Group 16 Application

ISQS 7339

Jonathan De Los Santos

Daniel Gerhart

Safwan Hasan

Case Study 3: Monte-Carlo Manufacturing

Daniel Gerhart

To develop a simulation model to determine how many machines of each type and number of employees that will be required to meet the forecasted demand I first loaded in the data to R studio.

In [125…
```r
suppressWarnings(library(dplyr))
suppressWarnings(library(tidyr))
mean_time <- data.frame(PartType = 1:6,
                        MachineA = c(3.5, 3.4, 1.8, 2.4, 4.2, 4),
                        MachineB = c(2.6, 2.5, 3.5, 5.8, 4.3, 4.3),
                        MachineC = c(8.9, 8, 12.6, 12.5, 28, 28))

std_dev <- data.frame(PartType = 1:6,
                      MachineA = c(0.15, 0.15, 0.1, 0.15, 0.15, 0.15),
                      MachineB = c(0.12, 0.12, 0.15, 0.15, 0.15, 0.15),
                      MachineC = c(0.15, 0.15, 0.25, 0.25, 0.5, 0.5))

demand <- data.frame(PartType = 1:6,
                     Demand = c(42, 18, 6, 6, 6, 6))
```

Then I calculated the total product time for each part type by:

In [126…
```r
total_time <- mean_time %>%
  gather(key = "Machine", value = "MeanTime", -PartType) %>%
  left_join(demand, by = "PartType") %>%
  mutate(TotalTime = MeanTime * Demand) %>%
  spread(key = "Machine", value = "TotalTime")

weekly_capacity <- 120 * 0.7 # 70% efficiency
machine_capacity <- weekly_capacity / total_time
```

Lastly I calculated the number of machines and employees required for production:

In [127…

```
weekly_capacity <- 120 * 0.7 # 70% efficiency
machine_capacity <- weekly_capacity / total_time

# Determine the number of machines required for each machine type
machines_required <- machine_capacity %>%
  mutate(MachineA = ceiling(MachineA),
         MachineB = ceiling(MachineB),
         MachineC = ceiling(MachineC))

# Determine the number of employees required for each machine type
employees_required <- machines_required %>%
  mutate(MachineA_Employees = MachineA / 2,
         MachineB_Employees = MachineB / 2,
         MachineC_Employees = MachineC)

employees_required
```

A data.frame: 18 × 9

PartType <dbl>	MeanTime <dbl>	Demand <dbl>	MachineA <dbl>	MachineB <dbl>	MachineC <dbl>	MachineA_Employees <dbl>	Ma
84.0	32.307692	2.000000	NA	1	NA	NA	
84.0	24.000000	2.000000	1	NA	NA	0.5	
84.0	9.438202	2.000000	NA	NA	1	NA	
42.0	33.600000	4.666667	NA	2	NA	NA	
42.0	24.705882	4.666667	2	NA	NA	1.0	
42.0	10.500000	4.666667	NA	NA	1	NA	
28.0	46.666667	14.000000	8	NA	NA	4.0	
28.0	24.000000	14.000000	NA	4	NA	NA	
28.0	6.666667	14.000000	NA	NA	2	NA	
21.0	35.000000	14.000000	6	NA	NA	3.0	
21.0	14.482759	14.000000	NA	3	NA	NA	
21.0	6.720000	14.000000	NA	NA	2	NA	
16.8	20.000000	14.000000	4	NA	NA	2.0	
16.8	19.534884	14.000000	NA	4	NA	NA	
16.8	3.000000	14.000000	NA	NA	1	NA	
14.0	21.000000	14.000000	4	NA	NA	2.0	
14.0	19.534884	14.000000	NA	4	NA	NA	
14.0	3.000000	14.000000	NA	NA	1	NA	

For the forecasted demand we will need 4 of machine A, 4 of machine B, 13 of machine C, and 17 employees.

Case Study 5: Emergency Room Discrete Event Simulation

Jonathan De Los Santos

The following case study models an emergency room with different system flows depending on the priority of incoming patients. First a distribution and its corresponding parameters must be decided from a historical dataset. Afterwards, we will move to modeling the ER simulation and proposing improvements to the process.

Case:

Patients arrive at the Emergency Room following an unknown probability distribution (stream 1). They will be treated by either of two doctors.

A proportion of the patients are classified as NIA (need immediate attention) and the rest as CW (can wait). NIA patients are given the highest priority, 3, see a doctor as soon as possible for 40 ± 30 minutes (stream 2), then have their priority reduced to 2 and wait until a doctor is free again, when they receive further treatment for 30 ± 20 minutes (stream 3) and are discharged.

CW patients initially receive a priority of 1 and are treated (when their turn comes) for 15 ± 10 minutes (stream 4); their priority is then increased to 2, they wait again until a doctor is free, receive 10 ± 5 minutes (stream 5) of final treatment and are discharged.

An important aspect of this system is that patients who have already seen the doctor once compete with newly arriving patients who need a doctor. As indicated, patients who have already seen the doctor once have a priority level of 2 (either increased from 1 to 2 or decreased from 3 to 2). Thus, there is one shared queue for the first treatment activity and the final treatment activity. In addition, we assume that the doctors are interchangeable. That is, it does not matter which of the two doctors performs the first or final treatment.

Simulate for 20 days of continuous operation, 24 hours per day. Note, the inter-arrival time and type of 100 patients are collected (based on a randomly selected weekday data). The data is attached.

1. *Analyze your results and explain your suggestions for reducing the waiting time of the patients.*

2. *What is the average flow-time for NIA and CW patients before or after applying suggestions. different suggestions.*

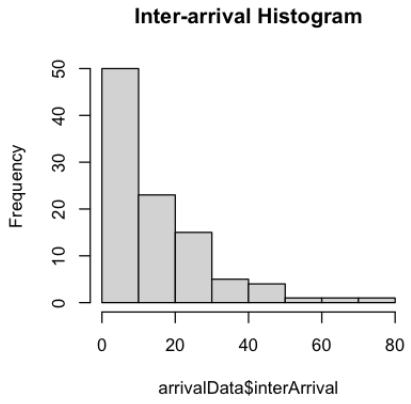
3. *Discuss the utilization of doctors before or after applying suggestions.*

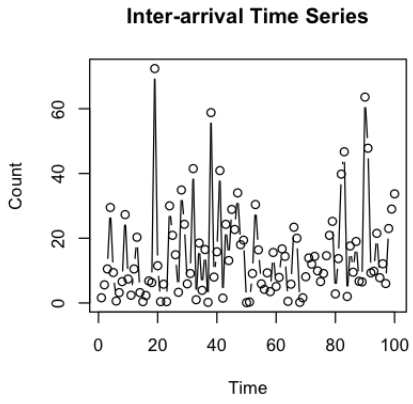
Data Input Analysis

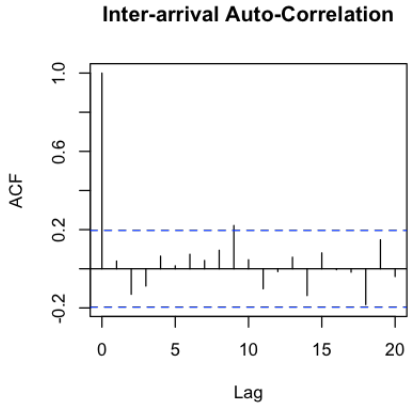
Before starting our discrete event simulation, we need to examine our patient arrival data and make a determination about its probability distribution.

In [1]:
```
arrivalData = read.csv("Data Sets/GroupCase5Data.csv")
```

In [10]:
```
suppressWarnings(library(gridExtra))
suppressWarnings(library(simmer.plot))

# Set up a 2x2 plot
par(mfrow = c(2, 2))

hist(arrivalData$interArrival, main="Inter-arrival Histogram")
plot(arrivalData$interArrival, type="b", main="Inter-arrival Time Series", y
acf(arrivalData$interArrival, main= "Inter-arrival Auto-Correlation")

# Reset the layout to default
par(mfrow = c(1, 1))
```



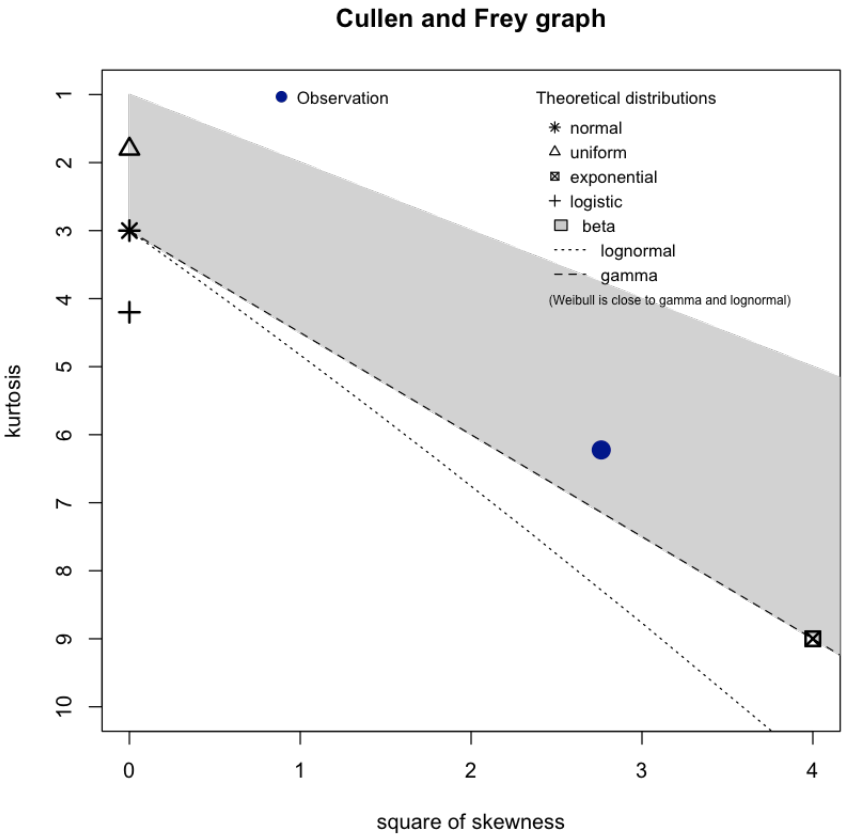




We can examine the data using the `fitdistrplus` library. In this instance, we will assume our arrival data is continuuous (`discrete = FALSE`).

In [12]:
```
suppressWarnings(library(fitdistrplus))
descdist(arrivalData$interArrival, discrete = FALSE)
```

```
summary statistics
------
min:  0.1   max:  72.4
median:  10.2
mean:  15.077
estimated sd:  14.35904
estimated skewness:  1.661798
estimated kurtosis:  6.224021
```



Based on our Cullen and Frey graph, we will examine lognormal, gamma, and exponential distributions.

In [15]:
```
fit.exp = fitdist(arrivalData$interArrival, "exp")
fit.gamma = fitdist(arrivalData$interArrival, "gamma")
fit.lnorm = fitdist(arrivalData$interArrival, "lnorm")
```

In [16]:
```
summary(fit.exp)
gofstat(fit.exp)
summary(fit.gamma)
gofstat(fit.gamma)
summary(fit.lnorm)
gofstat(fit.lnorm)
```

```
Fitting of the distribution ' exp ' by maximum likelihood
Parameters :
       estimate  Std. Error
rate 0.06632619 0.006631111
Loglikelihood:  -371.317   AIC:  744.6341   BIC:  747.2393
Goodness-of-fit statistics
                               1-mle-exp
Kolmogorov-Smirnov statistic 0.07025042
Cramer-von Mises statistic   0.04728725
Anderson-Darling statistic   0.40070355

Goodness-of-fit criteria
                                1-mle-exp
Akaike's Information Criterion  744.6341
Bayesian Information Criterion  747.2393
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
        estimate Std. Error
shape 0.96354630 0.11955644
rate  0.06388923 0.01025346
Loglikelihood:  -371.2718   AIC:  746.5435   BIC:  751.7539
Correlation matrix:
          shape      rate
shape 1.0000000 0.7728234
rate  0.7728234 1.0000000
Goodness-of-fit statistics
                              1-mle-gamma
Kolmogorov-Smirnov statistic  0.07804184
Cramer-von Mises statistic    0.05957576
Anderson-Darling statistic    0.41550172

Goodness-of-fit criteria
                               1-mle-gamma
Akaike's Information Criterion    746.5435
Bayesian Information Criterion    751.7539
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
        estimate Std. Error
meanlog 2.111460 0.13654252
sdlog   1.365425 0.09654991
Loglikelihood:  -384.1864   AIC:  772.3729   BIC:  777.5832
Correlation matrix:
        meanlog sdlog
meanlog       1     0
sdlog         0     1
```

```
Goodness-of-fit statistics
                                1-mle-lnorm
Kolmogorov-Smirnov statistic   0.1479491
Cramer-von Mises statistic     0.4586617
Anderson-Darling statistic     2.8735691

Goodness-of-fit criteria
                                1-mle-lnorm
Akaike's Information Criterion    772.3729
Bayesian Information Criterion    777.5832
```

By comparing the Akaike's and Bayesion Information Criteria as well as the Loglikelihoods, it appears the data is best fit by an exponential distribution.

The rate (λ) for our exponential distribution is ~0.0663 which represents how many arrivals per minute we can expect at the ER. **The recipocral of this parameter ($\frac{1}{λ}$) gives us the estimated inter-arrival time of 15.1 minutes.**

Now we need to determine the probabilities of each patient classification from the data set so we can set up our trajectories accordingly.

In [26]:
```r
# Count the number of NIA and CW patients
nia_count <- sum(arrivalData$type == "NIA")
cw_count <- sum(arrivalData$type == "CW")

# Calculate probabilities
total_patients <- nrow(arrivalData)
nia_prob <- nia_count / total_patients
cw_prob <- cw_count / total_patients

cat("NIA Patient Probability: ", nia_prob, "\n")
cat("CW Patient Probability: ", cw_prob)
```

```
NIA Patient Probability:  0.18
CW Patient Probability:  0.82
```

Initial DES Model

In [83]:
```r
suppressWarnings(library(simmer))

set.seed(123)

nia_prob <- 0.25
cw_prob <- 0.75

envs <- lapply(1:20, function(i) {
  env <- simmer("ER") %>%
    add_resource("doctor", 2)

  patient <- trajectory("Patient Path") %>%
```

```
    branch(function()
      sample(1:2, size=1, replace=TRUE, prob=c(nia_prob,cw_prob)), continue=

      # NIA Patient
      trajectory("Needs Immediate Assistance") %>%
        set_attribute("priority", 3) %>%
        set_attribute("patientType", -3) %>%
        set_prioritization(c(3, 3, T)) %>%

        # NIA sees doctor for 10-70 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 70)) %>%
        release("doctor", 1) %>%

        # Priority is reduced
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 10-50 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 50)) %>%
        release("doctor", 1),

      # CW Patient
      trajectory("Can Wait") %>%
        set_attribute("priority", 1) %>%
        set_attribute("patientType", -1) %>%
        set_prioritization(c(1, 3, T)) %>%

        # CW sees doctor for 5-25 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 25)) %>%
        release("doctor", 1) %>%

        # Priority is raised
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 5-15 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 15)) %>%
        release("doctor", 1)
    )
  env %>%
    add_generator("patient", patient, function() rexp(1,1/15), mon = 2)

  env %>%
    # Simulate 24 hour day in minutes
    run(1440)
})
```

Initial Model Results

The initial model with its two doctors reveals long flow and wait times for both NIA and CW patients. CW patients feel the brunt of this pain, with nearly twice as long of a flow time that is mostly spent waiting.

Looking at our plots for waiting and flow time, we see that these both continuosly increase over time showing that the system perpetually backs up. This is especially evident in our resource usage graph, which shows over all simulations that both doctors remain at max utilization after about halfway through the day.

In [115…

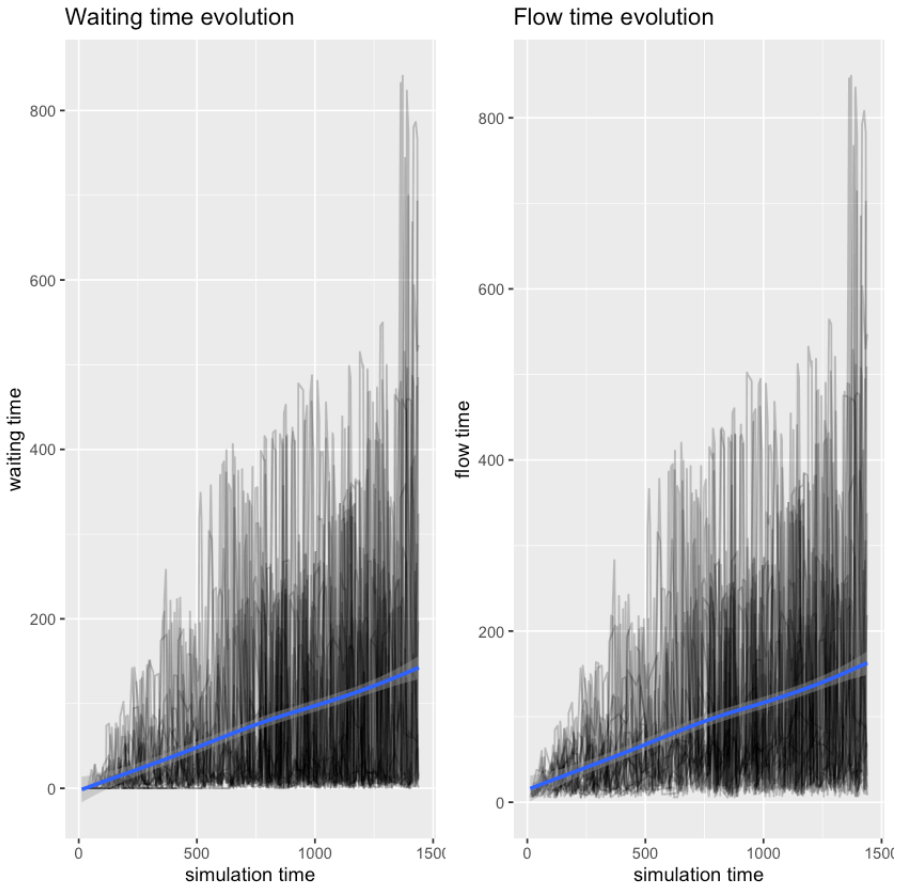
```
x1 <- get_mon_arrivals(envs)
x2 <- get_mon_attributes(envs)
all <- merge(x1, x2, by=c("name", "replication"), all = T)
all <- na.omit(all)

# High Priority flow and wait time
NIA <- subset(all, all$value == 3)
nia.flowTime = (NIA$end_time-NIA$start_time)
nia.waitTime = (NIA$end_time - NIA$start_time) - NIA$activity_time

# Low Priority flow and wait time
CW <- subset(all, all$value == 1)
cw.flowTime = (CW$end_time-CW$start_time)
cw.waitTime = (CW$end_time - CW$start_time) - CW$activity_time

cat("NIA Priority Average Flow Time: ", mean(nia.flowTime, na.rm = T), "\n")
cat("NIA Priority Average Wait Time: ", mean(nia.waitTime, na.rm = T), "\n")
cat("CW Priority Average Flow Time: ", mean(cw.flowTime, na.rm = T), "\n")
cat("CW Priority Average Wait Time: ", mean(cw.waitTime, na.rm = T))
```

```
NIA Priority Average Flow Time:  108.1191
NIA Priority Average Wait Time:  37.8005
CW Priority Average Flow Time:  206.1234
CW Priority Average Wait Time:  181.075
```

In [104…

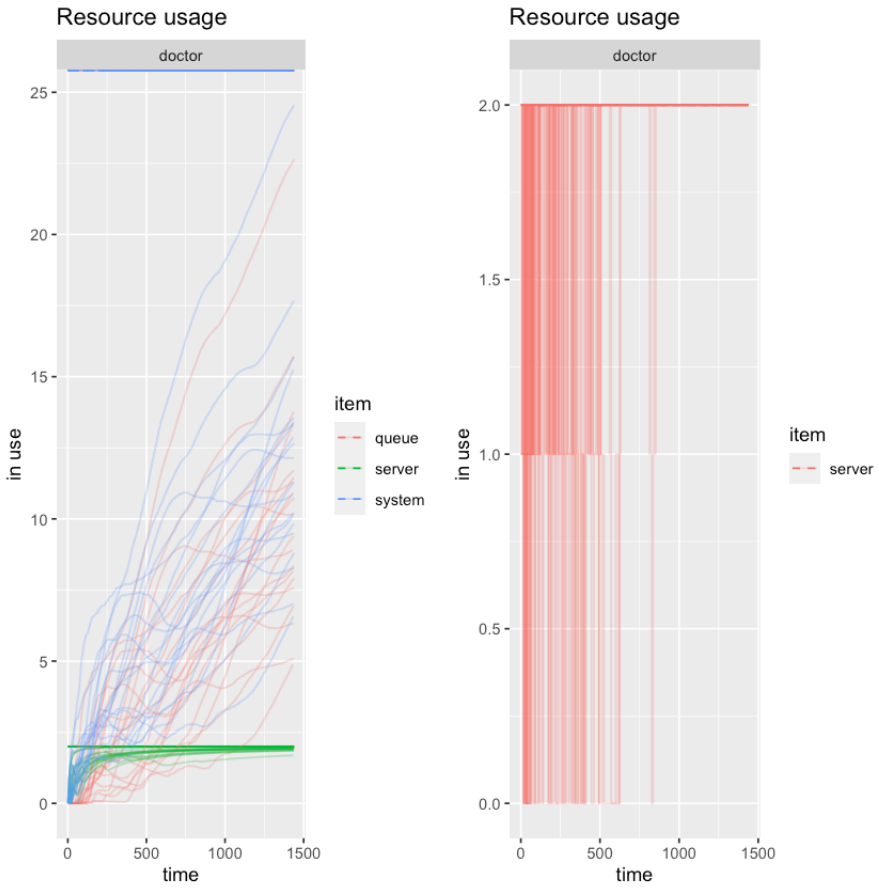
```
suppressWarnings(library(gridExtra))

arrivals <- get_mon_arrivals(envs, per_resource = T)

p1 <- plot(arrivals, metric = "waiting_time")
p2 <- plot(arrivals, metric = "flow_time")

grid.arrange(p1,p2, ncol=2)
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In [105…

```
resources <- get_mon_resources(envs)
p3 <- plot(resources, metric = "usage")
p4 <- plot(get_mon_resources(envs), metric = "usage", "doctor", items = "ser

grid.arrange(p3,p4, ncol=2)
```



DES Iteration 1: Add a Doctor

First we will attempt an obvious solution, adding the additional resource of one doctor.

In [113…

```
suppressWarnings(library(simmer))

set.seed(123)

nia_prob <- 0.25
cw_prob <- 0.75

envs2 <- lapply(1:20, function(i) {
  env2 <- simmer("ER") %>%
    add_resource("doctor", 3)

  patient <- trajectory("Patient Path") %>%

    branch(function()
      sample(1:2, size=1, replace=TRUE, prob=c(nia_prob,cw_prob)), continue=

      # NIA Patient
      trajectory("Needs Immediate Assistance") %>%
        set_attribute("priority", 3) %>%
        set_attribute("patientType", -3) %>%
        set_prioritization(c(3, 3, T)) %>%
```

```
        # NIA sees doctor for 10-70 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 70)) %>%
        release("doctor", 1) %>%

        # Priority is reduced
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 10-50 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 50)) %>%
        release("doctor", 1),

      # CW Patient
      trajectory("Can Wait") %>%
        set_attribute("priority", 1) %>%
        set_attribute("patientType", -1) %>%
        set_prioritization(c(1, 3, T)) %>%

        # CW sees doctor for 5-25 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 25)) %>%
        release("doctor", 1) %>%

        # Priority is raised
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 5-15 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 15)) %>%
        release("doctor", 1)
    )
  env2 %>%
    add_generator("patient", patient, function() rexp(1,1/15), mon = 2)

  env2 %>%
    # Simulate 24 hour day in minutes
    run(1440)
})
```

Iteration 1 Results

Adding a single doctor dramatically improved flow and wait times for both NIA and CW patients. NIA patients still have higher flow times due to their longer doctor visits, but their wait time is now a very small part of their overall activity time. CW patients still spend about half their time waiting, but their flow time is now actually less than NIA patients.

The waiting and flow time plots show that they are no longer constantly increasing throughout the day. After an initial spike there is a dip in both times, although it is unclear if the trajectory would continue upwards after a single day. The resource usage plots do seem to indicate that usage continues to decrease through the end of the day, and the doctors no longer remain at constant max utilization.

In [114…

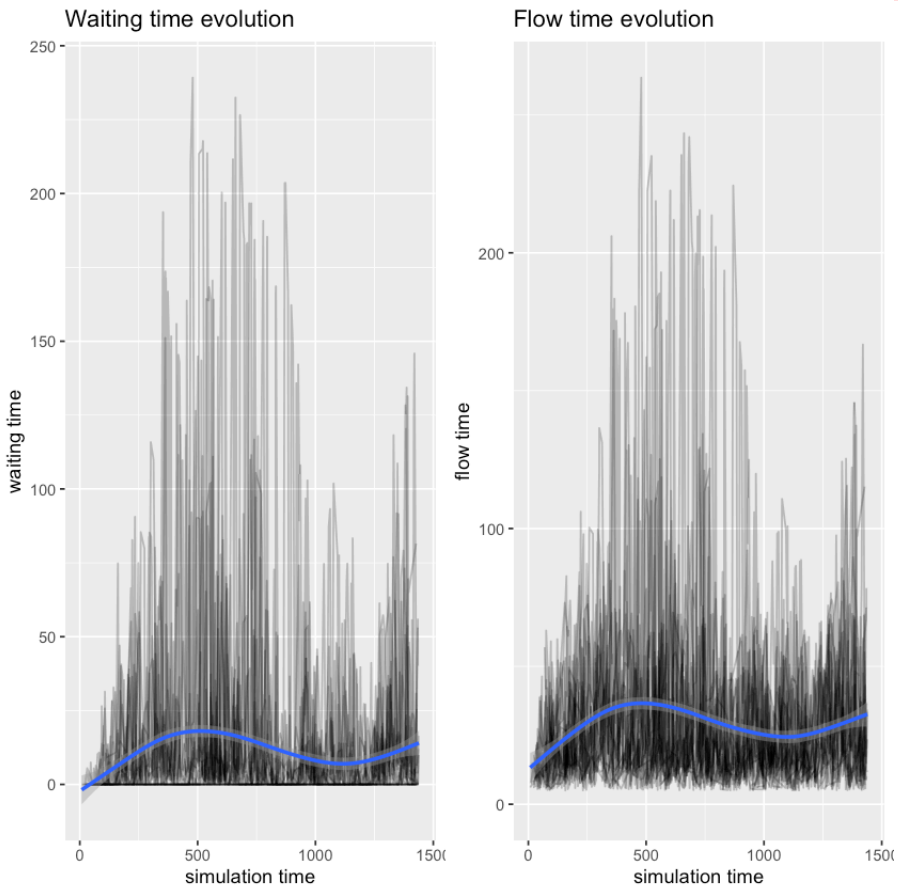
```
x1 <- get_mon_arrivals(envs2)
x2 <- get_mon_attributes(envs2)
all <- merge(x1, x2, by=c("name", "replication"), all = T)
all <- na.omit(all)

# High Priority flow and wait time
NIA <- subset(all, all$value == 3)
nia.flowTime = (NIA$end_time-NIA$start_time)
nia.waitTime = (NIA$end_time - NIA$start_time) - NIA$activity_time

# Low Priority flow and wait time
CW <- subset(all, all$value == 1)
cw.flowTime = (CW$end_time-CW$start_time)
cw.waitTime = (CW$end_time - CW$start_time) - CW$activity_time

cat("NIA Priority Average Flow Time: ", mean(nia.flowTime, na.rm = T), "\n")
cat("NIA Priority Average Wait Time: ", mean(nia.waitTime, na.rm = T), "\n")
cat("CW Priority Average Flow Time: ", mean(cw.flowTime, na.rm = T), "\n")
cat("CW Priority Average Wait Time: ", mean(cw.waitTime, na.rm = T))
```

```
NIA Priority Average Flow Time:  77.03679
NIA Priority Average Wait Time:  8.458684
CW Priority Average Flow Time:  52.24723
CW Priority Average Wait Time:  27.27617
```

In [108…

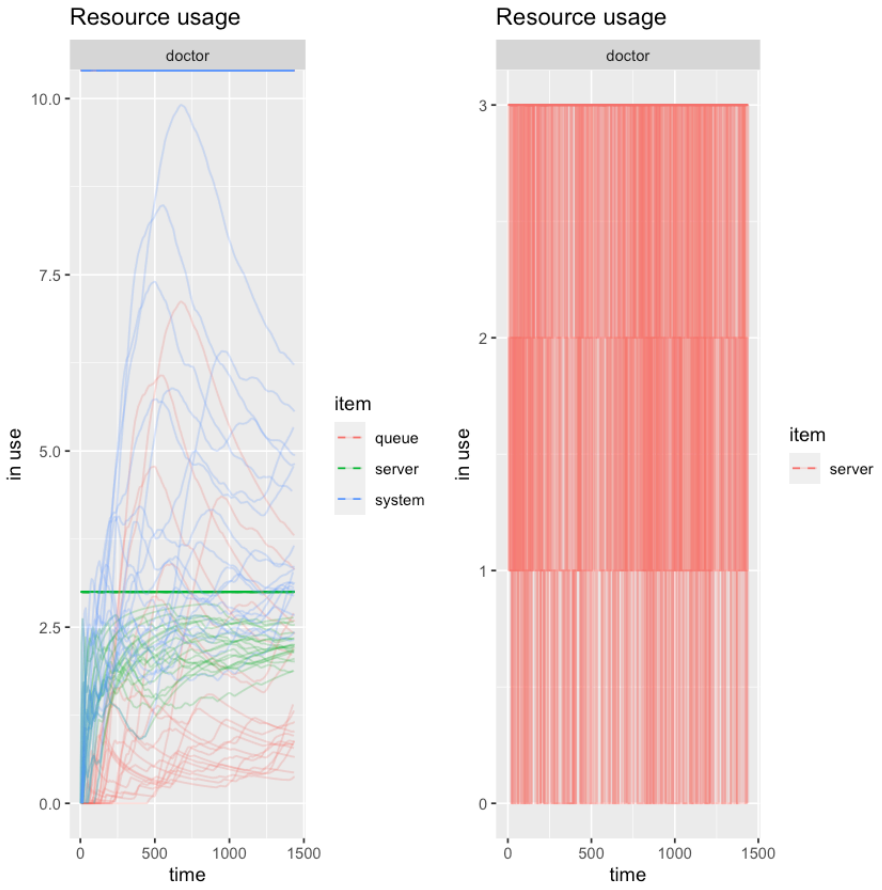
```
suppressWarnings(library(gridExtra))

arrivals <- get_mon_arrivals(envs2, per_resource = T)

p5 <- plot(arrivals, metric = "waiting_time")
p6 <- plot(arrivals, metric = "flow_time")

grid.arrange(p5,p6, ncol=2)
```

```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In [110…

```
resources <- get_mon_resources(envs2)
p7 <- plot(resources, metric = "usage")
p8 <- plot(get_mon_resources(envs2), metric = "usage", "doctor", items = "se

grid.arrange(p7,p8, ncol=2)
```



DES Iteration 2: Doctors Grow on Trees

One element of this model that sticks out is the dramatically higher seize times of doctors by NIA patients. While improvements to flow and wait times could be improved by reducing those times, I prefer that hospitals increase resources rather than reduce the quality of care. Hospital shareholders hate this one weird trick, but they can build their own simulation models.

This iteration adds an additional doctor resource. It's not very creative, but it is quite effective.

In [120…

```
suppressWarnings(library(simmer))

set.seed(123)

nia_prob <- 0.25
cw_prob <- 0.75

envs3 <- lapply(1:20, function(i) {
  env3 <- simmer("ER") %>%
    add_resource("doctor", 4)

  patient <- trajectory("Patient Path") %>%

    branch(function()
```

```
      sample(1:2, size=1, replace=TRUE, prob=c(nia_prob,cw_prob)), continue=

      # NIA Patient
      trajectory("Needs Immediate Assistance") %>%
        set_attribute("priority", 3) %>%
        set_attribute("patientType", -3) %>%
        set_prioritization(c(3, 3, T)) %>%

        # NIA sees doctor for 10-70 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 70)) %>%
        release("doctor", 1) %>%

        # Priority is reduced
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 10-50 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 10, 50)) %>%
        release("doctor", 1),

      # CW Patient
      trajectory("Can Wait") %>%
        set_attribute("priority", 1) %>%
        set_attribute("patientType", -1) %>%
        set_prioritization(c(1, 3, T)) %>%

        # CW sees doctor for 5-25 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 25)) %>%
        release("doctor", 1) %>%

        # Priority is raised
        set_attribute("priority", 2) %>%
        set_prioritization(c(2, 3, T)) %>%

        # Received further treatment for 5-15 minutes
        seize("doctor", 1) %>%
        timeout(function() runif(1, 5, 15)) %>%
        release("doctor", 1)
    )
  env3 %>%
    add_generator("patient", patient, function() rexp(1,1/15), mon = 2)

  env3 %>%
    # Simulate 24 hour day in minutes
    run(1440)
})
```

Iteration 2 results

The low priority (CW) patients are the winners of adding a fourth doctor as their flow time is nearly halved. However, wait times are now extremely quick for all patients and not increasing over time. The resource usage graphs are now more populated around the middle of the plots rather than the top, suggesting that doctors spend less time fully utilized.

Perhaps this is not ideal in business school terms, but for an ER I would suggest it is beneficial to invest in lower wait times and not optimize solely for max utilization of physicians.

In [119…

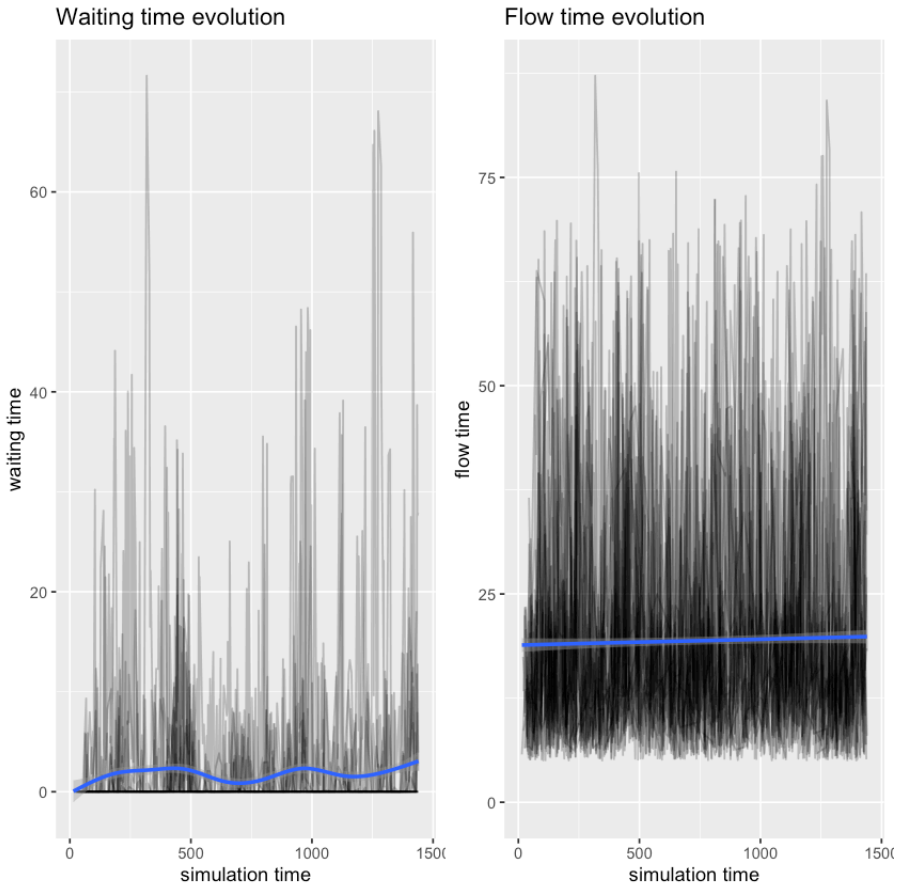
```
x1 <- get_mon_arrivals(envs3)
x2 <- get_mon_attributes(envs3)
all <- merge(x1, x2, by=c("name", "replication"), all = T)
all <- na.omit(all)

# High Priority flow and wait time
NIA <- subset(all, all$value == 3)
nia.flowTime = (NIA$end_time-NIA$start_time)
nia.waitTime = (NIA$end_time - NIA$start_time) - NIA$activity_time

# Low Priority flow and wait time
CW <- subset(all, all$value == 1)
cw.flowTime = (CW$end_time-CW$start_time)
cw.waitTime = (CW$end_time - CW$start_time) - CW$activity_time

cat("NIA Priority Average Flow Time: ", mean(nia.flowTime, na.rm = T), "\n")
cat("NIA Priority Average Wait Time: ", mean(nia.waitTime, na.rm = T), "\n")
cat("CW Priority Average Flow Time: ", mean(cw.flowTime, na.rm = T), "\n")
cat("CW Priority Average Wait Time: ", mean(cw.waitTime, na.rm = T))
```

```
NIA Priority Average Flow Time:  71.74587
NIA Priority Average Wait Time:  2.122879
CW Priority Average Flow Time:  28.76042
CW Priority Average Wait Time:  3.914827
```

In [121…

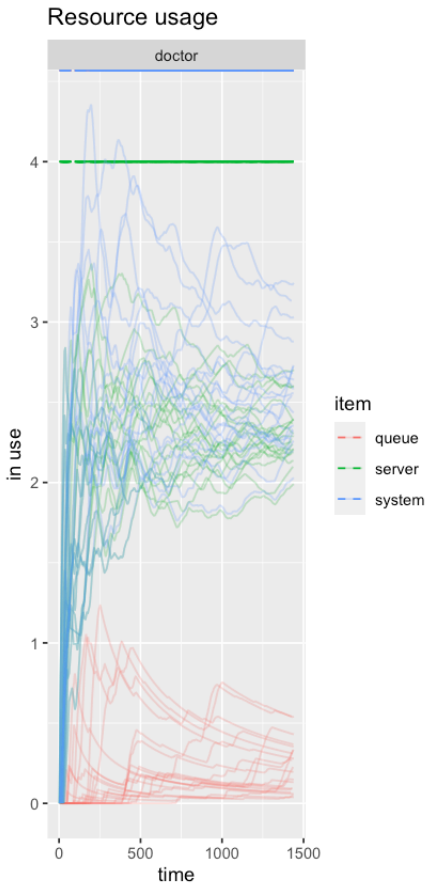
```
suppressWarnings(library(gridExtra))

arrivals <- get_mon_arrivals(envs3, per_resource = T)

p9 <- plot(arrivals, metric = "waiting_time")
p10 <- plot(arrivals, metric = "flow_time")

grid.arrange(p9,p10, ncol=2)
```

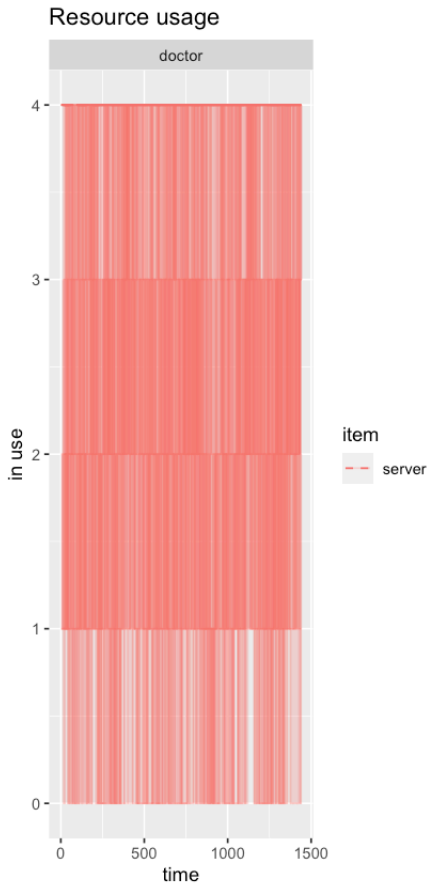
```
`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

`geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In [123…

```
resources <- get_mon_resources(envs3)
p11 <- plot(resources, metric = "usage")
p12 <- plot(get_mon_resources(envs3), metric = "usage", "doctor", items = "s

grid.arrange(p11,p12, ncol=2)
```


Resource usage
doctor
in use
time
item
queue
server
system


Resource usage
doctor
in use
time
item
server

Case Study 1: Capacity Concerns (Optimization)

Ryan Miller, either the founder or the CEO of CommuniCorp, is stressed about the financials of his business. CommuniCorp, the manufacturer of pagers, stocks plummeted within the last 12 months, losing about 25% of the pager market. The ticker price is also at an all-time low in the previous 52 weeks. Mr. Miller found that it was due to the lack of transparency and communication between the marketing and warehouse executives. He turned to the head of Corporate Information Management, Emily Jones, to request her to help him get his company back on track since CommuniCorp was behind on orders, even though their warehouse had adequate inventory.

(a) Emily Jones first decided to evaluate the number and type of servers to purchase on a month-to-month basis. Here, the **key parameter** is the number of employees in the company, which is 330, who need access to the intranet. An IP model to determine which servers Emily should purchase in a particular month to minimize costs in that month and support all the new users within 5 months, assuming there are 3 server manufacturers (Intel Pentium, SGI, and Sun):

Month 1:

Since we are limited in the budget to purchase a server in the first two months, we cannot spend over $9500. SGI is also willing to give us a discount of 10 percent off each server purchased, but only if purchased in the first or second month. So SGI Workstation for month 1 would be $9000 and can support all Sales department employees by month 2, and also a few of the manufacturing department employees. Sun, on the other hand, is willing to give us a 25 percent discount on all servers purchased in the first two months but that is unaffordable since $18750 is still higher than the $9500 budget for the first two months. So we will not buy any servers from Sun manufacturer. Therefore, the model is as follows:

Decision variables:

- x1-> the number of Standard Intel Pentium PC servers purchased in month 1
- x2 -> the number of Enhanced Intel Pentium PC servers purchased in month 1
- x3 -> the number of SGI Workstation servers purchased in month 1
- x4 -> the number of Sun Workstation servers purchased in month 1

Constraints:

- The total cost of servers purchased in month 1 cannot exceed $9,500:
  2500*x1 + 5000*x2+ 9000*x3 + 18750*x4 <= $9500

Objective function:

Maximize efficiency: 2500*x1 + 5000*x2+ 9000*x3 + 18750*x4

Model in Python:

```
import pulp
from pulp import *

# Create the problem
prob = LpProblem("Capacity concerns", LpMaximize)

# Define the decision variables
x1 = LpVariable("Standard", lowBound=0, cat = "Integer")
x2 = LpVariable("Enhanced", lowBound=0, cat = "Integer")
x3 = LpVariable("SGI", lowBound=0, cat = "Integer")
x4 = LpVariable("Sun", lowBound=0, cat = "Integer")

# Define the objective function
prob += 2500*x1 + 5000*x2+ 9000*x3 + 18750*x4

# Define the constraints
prob += 2500*x1 + 5000*x2+ 9000*x3 + 18750*x4 <= 9500


# Solve the problem
prob.solve()

# Print the solution status
print("Status:", LpStatus[prob.status])

# Print the optimal solution
print("Optimal Solution:")
print("Standard =", value(x1))
print("Enhanced =", value(x2))
print("SGI =", value(x3))
print("Sun=", value(x4))
print("Optimal Cost =", value(prob.objective))

print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Status: Optimal
Optimal Solution:
Standard = 0.0
Enhanced = 0.0
```

```
SGI = 1.0
Sun= 0.0
Optimal Cost = 9000.0
Sensitivity Analysis:
_C1 Shadow Price: -0.0 Slack: 500.0
```

According to the results, Emily should purchase 1 SGI workstation since that is the optimal number of servers within an optimal budget of $9000 to support all Sales department employees by month 2, and also a few of the manufacturing department employees by month 3.

The shadow price for constraint C1 is -0.0, which means that the optimal cost does not change if the right-hand side of constraint C1 changes. The slack for constraint C1 is 500.0, which means that the production capacity for product SGI has a surplus of 500 units, and the company can increase the demand for any of the other products by up to 500 units without affecting the optimal solution.

Month 2:
The server purchased in month 1 supports 200 employees, so all 50 Sales employees are supported and can already start to use the intranet. However, she is now left with a budget of $500, which is not enough to buy any servers. Also, since we used SGI's discount on month 1, we will have to pay the entire $10,000 for another SGI server. Let's still look at what our output might be with such complicated constraints:

Decision variables:

- x1-> the number of Standard Intel Pentium PC servers purchased in month 2
- x2 -> the number of Enhanced Intel Pentium PC servers purchased in month 2
- x3 -> the number of SGI Workstation servers purchased in month 2
- x4 -> the number of Sun Workstation servers purchased in month 2

Constraints:

- The total cost of servers purchased in month 2 cannot exceed $500: 2500*x1 + 5000*x2 + 10000*x3 + 18750*x4 <= $500

Objective function:
Maximize efficiency: 2500*x1 + 5000*x2 + 10000*x3 + 18750*x4

Model in Python:

```
import pulp
from pulp import *

# Create the problem
prob = LpProblem("Capacity concerns", LpMaximize)

# Define the decision variables
```

```
x1 = LpVariable("Standard", lowBound=0, cat = "Integer")
x2 = LpVariable("Enhanced", lowBound=0, cat = "Integer")
x3 = LpVariable("SGI", lowBound=0, cat = "Integer")
x4 = LpVariable("Sun", lowBound=0, cat = "Integer")

# Define the objective function
prob += 2500*x1 + 5000*x2+ 10000*x3 + 18750*x4

# Define the constraints
prob += 2500*x1 + 5000*x2+ 10000*x3 + 18750*x4 <= 500


# Solve the problem
prob.solve()

# Print the solution status
print("Status:", LpStatus[prob.status])

# Print the optimal solution
print("Optimal Solution:")
print("Standard =", value(x1))
print("Enhanced =", value(x2))
print("SGI =", value(x3))
print("Sun=", value(x4))
print("Optimal Cost =", value(prob.objective))

print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Status: Optimal
Optimal Solution:
Standard = 0.0
Enhanced = 0.0
SGI = 0.0
Sun= 0.0
Optimal Cost = 0.0
Sensitivity Analysis:
_C1 Shadow Price: -0.0 Slack: 500.0
```

According to the results, it also shows CommuniCorp cannot buy any server because of the budget constraint.

The sensitivity analysis shows that the shadow price (dual value) for constraint C1 is -0.0, which means that a one-unit increase in the right-hand side of constraint C1 would lead to a decrease of 0.0 in the optimal cost. The slack for C1 is 500.0, which means that the current resource level for constraint C1 is 500 units below the maximum capacity.

Month 3:

By now, we currently have an SGI Workstation server that already supports up to 200 employees. 50 of the Sales department employees and 150 out of the 180 Manufacturing employees are already covered. So for month 3, we would need to purchase a server to cover the remaining 30 Manufacturing employees while keeping the costs down (not necessarily a constraint that needs to be defined). We also have a new constraint: the manufacturing employees need three of the more powerful servers to communicate using the intranet. This means that we cannot purchase the Standard Intel Pentium PC server. We also lost the 25% off of Sun Workstation servers since it's already month 3, and have to pay the full price of $25000 for a Sun Workstation server. We are not, however, constrained by the budget. So our model would be as follows:

Decision variables:

- x1-> the number of Standard Intel Pentium PC servers purchased in month 3
- x2 -> the number of Enhanced Intel Pentium PC servers purchased in month 3
- x3 -> the number of SGI Workstation servers purchased in month 3
- x4 -> the number of Sun Workstation servers purchased in month 3

Constraints:

- At least one of the 3 more powerful servers: x2 + x3 + x4 >= 1
- Cover at least 30 of Manufacturing department employees: 30*x1 + 80*x2 + 200*x3 + 2000*x4 >= 30

Objective function:
Minimize: 2500*x1 + 5000*x2 + 10000*x3 + 25000*x4

Model in Python:

```
import pulp
from pulp import *

# Create the problem
prob = LpProblem("Capacity concerns", LpMinimize)

# Define the decision variables
x1 = LpVariable("Standard", lowBound=0, cat = "Integer")
```

```
x2 = LpVariable("Enhanced", lowBound=0, cat = "Integer")
x3 = LpVariable("SGI", lowBound=0, cat = "Integer")
x4 = LpVariable("Sun", lowBound=0, cat = "Integer")


# Define the objective function
prob += 2500*x1 + 5000*x2+ 10000*x3 + 25000*x4

# Define the constraints
prob += x2 + x3 + x4 >= 1
prob += 30*x1 + 80*x2 + 200*x3 + 2000*x4 >= 30

# Solve the problem
prob.solve()

# Print the solution status
print("Status:", LpStatus[prob.status])

# Print the optimal solution
print("Optimal Solution:")
print("Standard =", value(x1))
print("Enhanced =", value(x2))
print("SGI =", value(x3))
print("Sun=", value(x4))
print("Optimal Cost =", value(prob.objective))

print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Status: Optimal
Optimal Solution:
Standard = 0.0
Enhanced = 1.0
SGI = 0.0
Sun= 0.0
Optimal Cost = 5000.0
Sensitivity Analysis:
_C1 Shadow Price: 0.0 Slack: -0.0
_C2 Shadow Price: 0.0 Slack: -50.0
```

According to the results, Emily should purchase 1 Enhanced Intel Pentium PC server since that is the optimal number of servers to support the remaining 30 employees from the Manufacturing department, all 30 employees from the Warehouse department, and 20 out of 70 employees from the Marketing department. So now, the remaining number of employees is 50 who do not need to use the intranet until month 5 and belong to the Marketing department.

In the sensitivity analysis results, the shadow price for constraint _C1 is 0.0, which means that the optimal solution is not sensitive to changes in the number of servers for the Standard type. The slack for constraint _C1 is also 0, which means that the constraint is met exactly.
The shadow price for constraint _C2 is also 0.0, which means that the optimal solution is not sensitive to changes in the number of servers for the Enhanced type. However, the slack for constraint _C2 is -50.0, which means that the number of Enhanced servers used in the optimal solution is 50 less than the upper limit of 50.

Month 4:

The server purchased in month 3 supports the remaining 30 employees from the Manufacturing department, all 30 employees from the Warehouse department, and 20 out of 70 employees from the Marketing department. The remaining 50 employees from the Marketing department do not need the intranet until month 5, and hence there are no constraints this month. So Emily shouldn't need to purchase any servers this month but let's still see an LP model:

Decision variables:

- x1-> the number of Standard Intel Pentium PC servers purchased in month 4
- x2 -> the number of Enhanced Intel Pentium PC servers purchased in month 4
- x3 -> the number of SGI Workstation servers purchased in month 4
- x4 -> the number of Sun Workstation servers purchased in month 4

Constraints: None

Objective function:
Minimize: 2500*x1 + 5000*x2 + 10000*x3 + 18750*x4

Model in Python:

```
import pulp
from pulp import *

# Create the problem
prob = LpProblem("Capacity concerns", LpMaximize)

# Define the decision variables
x1 = LpVariable("Standard", lowBound=0, cat = "Integer")
```

```
x2 = LpVariable("Enhanced", lowBound=0, cat = "Integer")
x3 = LpVariable("SGI", lowBound=0, cat = "Integer")
x4 = LpVariable("Sun", lowBound=0, cat = "Integer")

# Define the objective function
prob += 2500*x1 + 5000*x2+ 10000*x3 + 18750*x4

# Solve the problem
prob.solve()

# Print the solution status
print("Status:", LpStatus[prob.status])

# Print the optimal solution
print("Optimal Solution:")
print("Standard =", value(x1))
print("Enhanced =", value(x2))
print("SGI =", value(x3))
print("Sun=", value(x4))
print("Optimal Cost =", value(prob.objective))

print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Status: Optimal
Optimal Solution:
Standard = 0.0
Enhanced = 0.0
SGI = 0.0
Sun= 0.0
Optimal Cost = 0.0
```

According to the results, it also shows the optimal number of servers needed is 0.

Month 5:

We need to buy at least one server to support the remaining 50 employees from the Marketing department while keeping the costs down. So the model is as follows:

Decision variables:

- x1-> the number of Standard Intel Pentium PC servers purchased in month 5
- x2 -> the number of Enhanced Intel Pentium PC servers purchased in month 5
- x3 -> the number of SGI Workstation servers purchased in month 5
- x4 -> the number of Sun Workstation servers purchased in month 5

Constraints:

- Cover at least 50 Marketing department employees: 30*x1 + 80*x2 + 200*x3 + 2000*x4 >= 50

Objective function:
Minimize: 2500*x1 + 5000*x2 + 10000*x3 + 25000*x4

Model in Python:

```
import pulp
from pulp import *

# Create the problem
prob = LpProblem("Capacity concerns", LpMinimize)

# Define the decision variables
x1 = LpVariable("Standard", lowBound=0, cat = "Integer")
x2 = LpVariable("Enhanced", lowBound=0, cat = "Integer")
x3 = LpVariable("SGI", lowBound=0, cat = "Integer")
x4 = LpVariable("Sun", lowBound=0, cat = "Integer")


# Define the objective function
prob += 2500*x1 + 5000*x2+ 10000*x3 + 25000*x4

# Define the constraints
prob += 30*x1 + 80*x2 + 200*x3 + 2000*x4 >= 50

# Solve the problem
prob.solve()

# Print the solution status
print("Status:", LpStatus[prob.status])

# Print the optimal solution
print("Optimal Solution:")
print("Standard =", value(x1))
print("Enhanced =", value(x2))
print("SGI =", value(x3))
print("Sun=", value(x4))
print("Optimal Cost =", value(prob.objective))
```

```
print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Status: Optimal
Optimal Solution:
Standard = 2.0
Enhanced = 0.0
SGI = 0.0
Sun= 0.0
Optimal Cost = 5000.0
Sensitivity Analysis:
_C1 Shadow Price: 0.0 Slack: -10.0
```

According to the results, Emily needs to purchase two Standard Intel Pentium PC servers that cover 60 employees and we only needed to cover 50, all within an optimal budget of $5000.

The optimal solution indicates that only Standard servers are used, with a quantity of 2, and the cost of this solution is $5000. The sensitivity analysis shows that the objective function value is not sensitive to changes in the cost of the Standard server (constraint C1), as the shadow price is 0.0, meaning that the optimal cost does not change with a unit increase or decrease in the cost of the Standard server. However, there is a slack of -10.0, indicating that the number of Standard servers used is 2, which is 10 units below the maximum allowed quantity of 12 according to constraint C1.

To summarize, Emily needs to purchase 1 SGI Workstation in month 1, 1 Enhanced Intel Pentium PC server in month 3, and 2 Standard Intel Pentium PC servers in month 5 which costs a total of $19,000 altogether.

(b) Emily decides to take a different approach to this dilemma of capacity concerns. She, therefore, decides to evaluate the number and type of servers to purchase over the entire planning period. The model code is as follows:

```
from pulp import *

# Define the problem
problem = LpProblem("Server Purchase problem", LpMinimize)

# Define decision variables
x = {(month, server): LpVariable(f"x{month}{server}", lowBound=0,
cat="Integer") for month in range(1, 6) for server in range(1, 5)}
```

```
# Define objective function
objective_function = lpSum([
    2500 * x[month, 1] + 5000 * x[month, 2] + (9000 if month in [1,
2] else 10000) * x[month, 3] + (18750 if month in [1, 2] else 25000)
* x[month, 4]
    for month in range(1, 6)
])
problem += objective_function

# Define constraints
problem += x[1, 1] + x[1, 2] + x[1, 3] + x[1, 4] == 0, "Month 1
Constraint"
problem += lpSum([2500 * x[1, 1] + 5000 * x[1, 2] + 9000 * x[1, 3] +
18750 * x[1, 4], 2500 * x[2, 1] + 5000 * x[2, 2] + 10000 * x[2, 3] +
18750 * x[2, 4]]) <= 9500, "Budget Constraint"
problem += x[2, 1] + x[2, 2] + x[2, 3] + x[2, 4] >= 2, "Month 2
Constraint"  # Sales department: 50 employees
problem += x[3, 1] + x[3, 2] + x[3, 3] + x[3, 4] >= 6, "Month 3
Constraint"  # Manufacturing department: 180 employees
problem += x[4, 1] + x[4, 2] + x[4, 3] + x[4, 4] >= 1, "Month 4
Constraint"  # Warehouse: 30 employees
problem += x[5, 1] + x[5, 2] + x[5, 3] + x[5, 4] >= 3, "Month 5
Constraint"  # Marketing department: 70 employees
problem += x[3, 2] + x[3, 3] + x[3, 4] >= 1, "Manufacturing
Requirement"

# Solve the problem
problem.solve()

# Print the results
for month in range(1, 6):
    print(f"Month {month}:")
    for server in range(1, 5):
        print(f"Number of server type {server} to purchase:
{int(x[month, server].varValue)}")
print(f"Total cost: ${value(problem.objective)}")

print("Sensitivity Analysis:")
for name, c in prob.constraints.items():
    shadow_price = c.pi
    slack = c.slack
    print(name, "Shadow Price:", shadow_price, "Slack:", slack)
```

Output:

```
Month 1:
Number of server type 1 to purchase: 0
Number of server type 2 to purchase: 0
Number of server type 3 to purchase: 0
Number of server type 4 to purchase: 0
Month 2:
Number of server type 1 to purchase: 2
Number of server type 2 to purchase: 0
Number of server type 3 to purchase: 0
Number of server type 4 to purchase: 0
Month 3:
Number of server type 1 to purchase: 5
Number of server type 2 to purchase: 1
Number of server type 3 to purchase: 0
Number of server type 4 to purchase: 0
Month 4:
Number of server type 1 to purchase: 1
Number of server type 2 to purchase: 0
Number of server type 3 to purchase: 0
Number of server type 4 to purchase: 0
Month 5:
Number of server type 1 to purchase: 3
Number of server type 2 to purchase: 0
Number of server type 3 to purchase: 0
Number of server type 4 to purchase: 0
Total cost: $32500.0
Sensitivity Analysis:
_C1 Shadow Price: 0.0 Slack: -10.0
```

The sensitivity analysis indicates that the shadow price for the cost constraint is 0.0, meaning that a small increase in the cost of purchasing servers would not affect the optimal solution. The slack value for the cost constraint is -10.0, indicating that the cost constraint is binding, and the optimal solution uses all the available budget for purchasing servers. Therefore, any increase in the cost of servers beyond the current cost would require an increase in the available budget to maintain the optimal solution.

To summarize, Emily needs to purchase 2 SGI Workstation servers in month 2, 5 Standard Intel Pentium PC servers and 1 Enhanced Intel Pentium PC server in month 3, and 1 Standard Intel Pentium PC servers in month 4, and 3 Standard Intel Pentium PC servers in month 5, which costs a total of $32,500 altogether.

c) The reason why the answer differs between the two methods is that they have different planning horizons. The first method in part (a) involves Emily evaluating the number and type of servers to purchase on a monthly basis, resulting in separate decisions for each month. However, this approach leads to a suboptimal solution as it doesn't consider potential cost

savings or capacity requirements for the entire planning period. On the other hand, the second method in part (b) involves Emily evaluating the number and type of servers to purchase over the entire planning period. This approach allows her to make better-informed decisions that minimize the total cost while meeting the capacity requirements for all departments over the entire timeline.

d) Emily may have overlooked some costs in her problem formulation, such as regular maintenance and support costs, varying energy costs due to different power consumption of servers, depreciation costs, setup and installation costs, and training and onboarding costs for employees to use the new intranet system effectively.

e) CommuniCorp's intranet implementation might encounter additional concerns from various departments, including security, usability, integration, scalability, reliability, performance, customization, and flexibility. These concerns encompass protecting sensitive information, ensuring user-friendliness, integrating with existing software, accommodating future growth, providing reliable and efficient operations, and being flexible to meet different department needs.