Case Studies

Eric Weinberg, Shelby Chapa, Anthany Wingo

## Problem 1

The problem is that Mr. Miller's company CommuniCorp has had stocks drop to their lowest point in 52 weeks. The main cause of this issue is an increase of missed or late orders. He discovered that this is due to a lack of communication between sales, marketing, and manufacturing departments. In which there is an overproduction of pagers and misinformation about inventory levels. Mr. Miller decides to install company wide computer networks (intranet) that will give access to critical documents and increase communication. He will rollout these the changes in phases over 5 months starting with sales and ending with marketing. Mr. Miller then hired on Emily to help with purchasing support for the new servers. Their options are Standard Intel Pentium PC, Enhanced Intel Pentium PC, SGI Workstation, and Sun Workstation. We will evaluate the best options for the 5 month timeframe and cost effectiveness.

# Import the PuLP library  
from pulp import \*  
  
# Define the Model and set the optimization sense to minimize (LpMinimize)  
model = LpProblem("Intranet Installation", LpMinimize)  
  
# Define the decision variables as the number of servers to purchase for each month and type  
server\_A\_Jan = LpVariable('server\_A\_Jan', lowBound=0, cat='Integer')  
server\_B\_Jan = LpVariable('server\_B\_Jan', lowBound=0, cat='Integer')  
server\_C\_Jan = LpVariable('server\_C\_Jan', lowBound=0, cat='Integer')  
server\_D\_Jan = LpVariable('server\_D\_Jan', lowBound=0, cat='Integer')  
server\_E\_Jan = LpVariable('server\_E\_Jan', lowBound=0, cat='Integer')  
  
server\_A\_Feb = LpVariable('server\_A\_Feb', lowBound=0, cat='Integer')  
server\_B\_Feb = LpVariable('server\_B\_Feb', lowBound=0, cat='Integer')  
server\_C\_Feb = LpVariable('server\_C\_Feb', lowBound=0, cat='Integer')  
server\_D\_Feb = LpVariable('server\_D\_Feb', lowBound=0, cat='Integer')  
server\_E\_Feb = LpVariable('server\_E\_Feb', lowBound=0, cat='Integer')  
  
server\_A\_Mar = LpVariable('server\_A\_Mar', lowBound=0, cat='Integer')  
server\_B\_Mar = LpVariable('server\_B\_Mar', lowBound=0, cat='Integer')  
server\_C\_Mar = LpVariable('server\_C\_Mar', lowBound=0, cat='Integer')  
server\_D\_Mar = LpVariable('server\_D\_Mar', lowBound=0, cat='Integer')  
server\_E\_Mar = LpVariable('server\_E\_Mar', lowBound=0, cat='Integer')  
  
server\_A\_Apr = LpVariable('server\_A\_Apr', lowBound=0, cat='Integer')  
server\_B\_Apr = LpVariable('server\_B\_Apr', lowBound=0, cat='Integer')  
server\_C\_Apr = LpVariable('server\_C\_Apr', lowBound=0, cat='Integer')  
server\_D\_Apr = LpVariable('server\_D\_Apr', lowBound=0, cat='Integer')  
server\_E\_Apr = LpVariable('server\_E\_Apr', lowBound=0, cat='Integer')  
  
server\_A\_May = LpVariable('server\_A\_May', lowBound=0, cat='Integer')  
server\_B\_May = LpVariable('server\_B\_May', lowBound=0, cat='Integer')  
server\_C\_May = LpVariable('server\_C\_May', lowBound=0, cat='Integer')  
server\_D\_May = LpVariable('server\_D\_May', lowBound=0, cat='Integer')  
server\_E\_May = LpVariable('server\_E\_May', lowBound=0, cat='Integer')  
  
  
# Define the objective function to minimize the total cost  
model += 2500 \* (server\_A\_Jan + server\_A\_Feb + server\_A\_Mar + server\_A\_Apr + server\_A\_May) + \  
 5000 \* (server\_B\_Jan + server\_B\_Feb + server\_B\_Mar + server\_B\_Apr + server\_B\_May) + \  
 10000 \* (server\_C\_Jan + server\_C\_Feb + server\_C\_Mar + server\_C\_Apr + server\_C\_May) + \  
 25000 \* (server\_D\_Jan + server\_D\_Feb + server\_D\_Mar + server\_D\_Apr + server\_D\_May) + \  
 2500 \* (server\_E\_Jan + server\_E\_Feb + server\_E\_Mar + server\_E\_Apr + server\_E\_May), "Total Cost"  
  
# Define the constraints  
  
# Constraint 1: At least 50 employees should be supported by the server in month 2  
model += 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb >= 50  
  
# Constraint 2: At least 180 employees should be supported by the server in month 3  
model += 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar >= 180  
  
# Constraint 3: At least 30 employees should be supported by the server in month 4  
model += 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar + \  
 30 \* server\_A\_Apr + 80 \* server\_B\_Apr + 200 \* server\_C\_Apr + 2000 \* server\_D\_Apr + 30 \* server\_E\_Apr >= 30  
  
# Constraint 4: At least 70 employees should be supported by the server in month 5  
model += 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar + \  
 30 \* server\_A\_Apr + 80 \* server\_B\_Apr + 200 \* server\_C\_Apr + 2000 \* server\_D\_Apr + 30 \* server\_E\_Apr + \  
 70 \* server\_A\_May + 80 \* server\_B\_May + 200 \* server\_C\_May + 2000 \* server\_D\_May + 30 \* server\_E\_May >= 70, "Constraint 4"  
  
  
# Constraint 5: Total cost of servers in months 1 and 2 should be less than or equal to $9,500  
model += 2500 \* (server\_A\_Jan + server\_B\_Jan) + 5000 \* (server\_C\_Jan + server\_D\_Jan + server\_E\_Jan) + \  
 2500 \* (server\_A\_Feb + server\_B\_Feb) + 5000 \* (server\_C\_Feb + server\_D\_Feb + server\_E\_Feb) <= 9500  
  
# Constraint 6: Only one of server\_C should be equal to 1  
model += server\_C\_Jan + server\_C\_Feb + server\_C\_Mar + server\_C\_Apr + server\_C\_May == 1  
  
# Constraint 7: The number of employees in Sales cannot exceed the capacity of the server in Month 2  
model += 50 <= 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan  
model += 50 <= 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb  
  
# Constraint 8: The number of employees in Manufacturing cannot exceed the capacity of the server in Month 3  
model += 180 <= 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar  
  
# Constraint 9: The number of employees in Warehouse cannot exceed the capacity of the server in Month 4  
model += 30 <= 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar + \  
 30 \* server\_A\_Apr + 80 \* server\_B\_Apr + 200 \* server\_C\_Apr + 2000 \* server\_D\_Apr + 30 \* server\_E\_Apr  
  
# Constraint 10: The number of employees in Marketing cannot exceed the capacity of the server in Month 5  
model += 70 <= 30 \* server\_A\_Jan + 80 \* server\_B\_Jan + 200 \* server\_C\_Jan + 2000 \* server\_D\_Jan + 30 \* server\_E\_Jan + \  
 30 \* server\_A\_Feb + 80 \* server\_B\_Feb + 200 \* server\_C\_Feb + 2000 \* server\_D\_Feb + 30 \* server\_E\_Feb + \  
 30 \* server\_A\_Mar + 80 \* server\_B\_Mar + 200 \* server\_C\_Mar + 2000 \* server\_D\_Mar + 30 \* server\_E\_Mar + \  
 30 \* server\_A\_Apr + 80 \* server\_B\_Apr + 200 \* server\_C\_Apr + 2000 \* server\_D\_Apr + 30 \* server\_E\_Apr + \  
 70 \* server\_A\_May + 80 \* server\_B\_May + 200 \* server\_C\_May + 2000 \* server\_D\_May + 70 \* server\_E\_May  
  
# Solve the optimization problem  
status = model.solve()  
  
# Print the status of the solution  
print("Status:", LpStatus[status])  
  
# Print the optimized value of the objective function  
print("Total Cost =", value(model.objective))  
  
# Print the values of the decision variables  
for var in model.variables():  
 print(var.name, "=", var.varValue)

Result - Optimal solution found  
  
Objective value: 15000.00000000  
Enumerated nodes: 0  
Total iterations: 0  
Time (CPU seconds): 0.01  
Time (Wallclock seconds): 0.01  
  
Option for printingOptions changed from normal to all  
Total time (CPU seconds): 0.01 (Wallclock seconds): 0.01  
  
Status: Optimal  
Total Cost = 15000.0  
server\_A\_Apr = 0.0  
server\_A\_Feb = 0.0  
server\_A\_Jan = 0.0  
server\_A\_Mar = 0.0  
server\_A\_May = 0.0  
server\_B\_Apr = 0.0  
server\_B\_Feb = 1.0  
server\_B\_Jan = 0.0  
server\_B\_Mar = 0.0  
server\_B\_May = 0.0  
server\_C\_Apr = 0.0  
server\_C\_Feb = 0.0  
server\_C\_Jan = 1.0  
server\_C\_Mar = 0.0  
server\_C\_May = 0.0  
server\_D\_Apr = 0.0  
server\_D\_Feb = 0.0  
server\_D\_Jan = 0.0  
server\_D\_Mar = 0.0  
server\_D\_May = 0.0  
server\_E\_Apr = 0.0  
server\_E\_Feb = 0.0  
server\_E\_Jan = 0.0  
server\_E\_Mar = 0.0  
server\_E\_May = 0.0

The optimization problem presented in this case study aimed to identify the optimal assignment of servers to specific tasks in order to minimize costs while meeting certain requirements. The optimal solution was found with an objective value of 15000.0, indicating that costs were minimized. The results showed that servers B and C were assigned to tasks in the months of February and January, respectively, while the other servers were not assigned to any tasks

Part B

The previous code only considered the purchase of servers for each month separately and optimized the cost for each month. This modified code considers the purchase of servers over the entire planning period and optimizes the cost while supporting all new users.

from pulp import \*  
  
# Define the Model and set the optimization sense to minimize (LpMinimize)  
model = LpProblem("Server Purchasing Plan", LpMinimize)  
  
# Define the decision variables as the number of servers to purchase for each month and type  
servers\_Jan\_A = LpVariable("servers\_Jan\_A", lowBound=0, cat='Integer')  
servers\_Jan\_B = LpVariable("servers\_Jan\_B", lowBound=0, cat='Integer')  
servers\_Jan\_C = LpVariable("servers\_Jan\_C", lowBound=0, cat='Integer')  
servers\_Jan\_D = LpVariable("servers\_Jan\_D", lowBound=0, cat='Integer')  
servers\_Jan\_E = LpVariable("servers\_Jan\_E", lowBound=0, cat='Integer')  
  
servers\_Feb\_A = LpVariable("servers\_Feb\_A", lowBound=0, cat='Integer')  
servers\_Feb\_B = LpVariable("servers\_Feb\_B", lowBound=0, cat='Integer')  
servers\_Feb\_C = LpVariable("servers\_Feb\_C", lowBound=0, cat='Integer')  
servers\_Feb\_D = LpVariable("servers\_Feb\_D", lowBound=0, cat='Integer')  
servers\_Feb\_E = LpVariable("servers\_Feb\_E", lowBound=0, cat='Integer')  
  
servers\_Mar\_A = LpVariable("servers\_Mar\_A", lowBound=0, cat='Integer')  
servers\_Mar\_B = LpVariable("servers\_Mar\_B", lowBound=0, cat='Integer')  
servers\_Mar\_C = LpVariable("servers\_Mar\_C", lowBound=0, cat='Integer')  
servers\_Mar\_D = LpVariable("servers\_Mar\_D", lowBound=0, cat='Integer')  
servers\_Mar\_E = LpVariable("servers\_Mar\_E", lowBound=0, cat='Integer')  
  
servers\_Apr\_A = LpVariable("servers\_Apr\_A", lowBound=0, cat='Integer')  
servers\_Apr\_B = LpVariable("servers\_Apr\_B", lowBound=0, cat='Integer')  
servers\_Apr\_C = LpVariable("servers\_Apr\_C", lowBound=0, cat='Integer')  
servers\_Apr\_D = LpVariable("servers\_Apr\_D", lowBound=0, cat='Integer')  
servers\_Apr\_E = LpVariable("servers\_Apr\_E", lowBound=0, cat='Integer')  
  
servers\_May\_A = LpVariable("servers\_May\_A", lowBound=0, cat='Integer')  
servers\_May\_B = LpVariable("servers\_May\_B", lowBound=0, cat='Integer')  
servers\_May\_C = LpVariable("servers\_May\_C", lowBound=0, cat='Integer')  
servers\_May\_D = LpVariable("servers\_May\_D", lowBound=0, cat='Integer')  
servers\_May\_E = LpVariable("servers\_May\_E", lowBound=0, cat='Integer')  
  
# Define the objective function to minimize the total cost  
model += (2500\*servers\_Jan\_A + 5000\*servers\_Jan\_B + 10000\*servers\_Jan\_C + 25000\*servers\_Jan\_D + 2500\*servers\_Jan\_E +  
 2500\*servers\_Feb\_A + 5000\*servers\_Feb\_B + 10000\*servers\_Feb\_C + 25000\*servers\_Feb\_D + 2500\*servers\_Feb\_E +  
 2500\*servers\_Mar\_A + 5000\*servers\_Mar\_B + 10000\*servers\_Mar\_C + 25000\*servers\_Mar\_D + 2500\*servers\_Mar\_E +  
 2500\*servers\_Apr\_A + 5000\*servers\_Apr\_B + 10000\*servers\_Apr\_C + 25000\*servers\_Apr\_D + 2500\*servers\_Apr\_E +  
 2500\*servers\_May\_A + 5000\*servers\_May\_B + 10000\*servers\_May\_C + 25000\*servers\_May\_D + 2500\*servers\_May\_E), "Total Cost"  
  
# Define the constraints  
  
# Define the objective function to minimize the total cost  
model += (2500\*servers\_Jan\_A + 5000\*servers\_Jan\_B + 10000\*servers\_Jan\_C + 25000\*servers\_Jan\_D + 2500\*servers\_Jan\_E +  
 2500\*servers\_Feb\_A + 5000\*servers\_Feb\_B + 10000\*servers\_Feb\_C + 25000\*servers\_Feb\_D + 2500\*servers\_Feb\_E +  
 2500\*servers\_Mar\_A + 5000\*servers\_Mar\_B + 10000\*servers\_Mar\_C + 25000\*servers\_Mar\_D + 2500\*servers\_Mar\_E +  
 2500\*servers\_Apr\_A + 5000\*servers\_Apr\_B + 10000\*servers\_Apr\_C + 25000\*servers\_Apr\_D + 2500\*servers\_Apr\_E +  
 2500\*servers\_May\_A + 5000\*servers\_May\_B + 10000\*servers\_May\_C + 25000\*servers\_May\_D + 2500\*servers\_May\_E), "Total Cost"  
  
# Define the constraints  
  
# Constraint 1: At least 50 employees should be supported by the server in month 2  
model += (30 \* servers\_Jan\_A + 80 \* servers\_Jan\_B + 200 \* servers\_Jan\_C + 2000 \* servers\_Jan\_D + 30 \* servers\_Jan\_E +  
 30 \* servers\_Feb\_A + 80 \* servers\_Feb\_B + 200 \* servers\_Feb\_C + 2000 \* servers\_Feb\_D + 30 \* servers\_Feb\_E) >= 50  
  
# Constraint 2: At least 180 employees should be supported by the server in month 3  
model += (30 \* servers\_Jan\_A + 80 \* servers\_Jan\_B + 200 \* servers\_Jan\_C + 2000 \* servers\_Jan\_D + 30 \* servers\_Jan\_E +  
 30 \* servers\_Feb\_A + 80 \* servers\_Feb\_B + 200 \* servers\_Feb\_C + 2000 \* servers\_Feb\_D + 30 \* servers\_Feb\_E +  
 30 \* servers\_Mar\_A + 80 \* servers\_Mar\_B + 200 \* servers\_Mar\_C + 2000 \* servers\_Mar\_D + 30 \* servers\_Mar\_E) >= 180  
  
# Constraint 3: At least 30 employees should be supported by the server in month 4  
model += (30 \* servers\_Jan\_A + 80 \* servers\_Jan\_B + 200 \* servers\_Jan\_C + 2000 \* servers\_Jan\_D + 30 \* servers\_Jan\_E +  
 30 \* servers\_Feb\_A + 80 \* servers\_Feb\_B + 200 \* servers\_Feb\_C + 2000 \* servers\_Feb\_D + 30 \* servers\_Feb\_E +  
 30 \* servers\_Mar\_A + 80 \* servers\_Mar\_B + 200 \* servers\_Mar\_C + 2000 \* servers\_Mar\_D + 30 \* servers\_Mar\_E +  
 30 \* servers\_Apr\_A + 80 \* servers\_Apr\_B + 200 \* servers\_Apr\_C + 2000 \* servers\_Apr\_D + 30 \* servers\_Apr\_E) >= 30  
  
# Constraint 4: At least 80 employees should be supported by the server in month 5  
model += (30 \* servers\_Jan\_A + 80 \* servers\_Jan\_B + 200 \* servers\_Jan\_C + 2000 \* servers\_Jan\_D + 30 \* servers\_Jan\_E +  
 30 \* servers\_Feb\_A + 80 \* servers\_Feb\_B + 200 \* servers\_Feb\_C + 2000 \* servers\_Feb\_D + 30 \* servers\_Feb\_E +  
 30 \* servers\_Mar\_A + 80 \* servers\_Mar\_B + 200 \* servers\_Mar\_C + 2000 \* servers\_Mar\_D + 30 \* servers\_Mar\_E +  
 30 \* servers\_Apr\_A + 80 \* servers\_Apr\_B + 200 \* servers\_Apr\_C + 2000 \* servers\_Apr\_D + 30 \* servers\_Apr\_E +  
 30 \* servers\_May\_A + 80 \* servers\_May\_B + 200 \* servers\_May\_C + 2000 \* servers\_May\_D + 30 \* servers\_May\_E) >= 80  
  
# Solve the optimization problem  
status = model.solve()  
  
# Print the status of the solution  
print("Status:", LpStatus[status])  
  
# Print the optimized value of the objective function  
print("Total Cost =", value(model.objective))  
  
# Print the values of the decision variables  
for var in model.variables():  
 print(var.name, "=", var.varValue)

Result - Optimal solution found  
  
Objective value: 10000.00000000  
Enumerated nodes: 0  
Total iterations: 0  
Time (CPU seconds): 0.01  
Time (Wallclock seconds): 0.02  
  
Option for printingOptions changed from normal to all  
Total time (CPU seconds): 0.01 (Wallclock seconds): 0.03  
  
Status: Optimal  
Number of type A servers to purchase in month 1: 0.0  
Number of type B servers to purchase in month 1: 0.0  
Number of type C servers to purchase in month 1: 1.0  
Number of type D servers to purchase in month 1: 0.0  
Number of type E servers to purchase in month 1: 0.0  
Number of type A servers to purchase in month 2: 0.0  
Number of type B servers to purchase in month 2: 0.0  
Number of type C servers to purchase in month 2: 0.0  
Number of type D servers to purchase in month 2: 0.0  
Number of type E servers to purchase in month 2: 0.0  
Number of type A servers to purchase in month 3: 0.0  
Number of type B servers to purchase in month 3: 0.0  
Number of type C servers to purchase in month 3: 0.0  
Number of type D servers to purchase in month 3: 0.0  
Number of type E servers to purchase in month 3: 0.0  
Number of type A servers to purchase in month 4: 0.0  
Number of type B servers to purchase in month 4: 0.0  
Number of type C servers to purchase in month 4: 0.0  
Number of type D servers to purchase in month 4: 0.0  
Number of type E servers to purchase in month 4: 0.0  
Number of type A servers to purchase in month 5: 0.0  
Number of type B servers to purchase in month 5: 0.0  
Number of type C servers to purchase in month 5: 0.0  
Number of type D servers to purchase in month 5: 0.0  
Number of type E servers to purchase in month 5: 0.0  
Total Cost: $ 10000.0

Comparing the two results, we can see that Emily’s initial thought of purchasing a larger server in the initial months to support users in the final months makes sense. Therefore, the IP model’s solution is more cost-efficient and effective in supporting all new users.

There might be other costs associated with the server purchase that she is not accounting for. For example, there could be additional costs for upgrading or integrating the servers into the existing network infrastructure. Additionally, there could be costs associated with training employees.

The timing and relationship to other departments is another concern that various departments of CommuniCorp might have regarding the intranet upgrade. If one department gets the upgrade earlier than others, it could lead to some departments feeling left behind or disadvantaged. It could also lead to problems of interopobility if all departments are relying on one critical system. Additionally, the timing of the upgrade might impact the workflow and processes of other departments.

## Problem 3

Sturgill Manufacturing, Inc. is facing a challenge in predicting the number of machines and employees required for its planned production for the coming year. To address this issue, the company needs to develop a simulation model that takes into account the plant’s operating hours, shop efficiency, and the time required to produce each part on each machine.

# Total hours available per week  
total\_hours <- 120  
  
# Shop efficiency parameters  
efficiency\_mean <- 0.7  
efficiency\_sd <- 0.05  
  
# Part production time parameters  
mean\_time <- matrix(c(3.5, 2.6, 8.9,  
 3.4, 2.5, 8,  
 1.8, 3.5, 12.6,  
 2.4, 5.8, 12.5,  
 4.2, 4.3, 28,  
 4, 4.3, 28),   
 nrow = 6, ncol = 3, byrow = TRUE)  
  
sd\_time <- matrix(c(0.15, 0.12, 0.15,  
 0.15, 0.12, 0.15,  
 0.1, 0.15, 0.25,  
 0.15, 0.15, 0.25,  
 0.15, 0.15, 0.5,  
 0.15, 0.15, 0.5),   
 nrow = 6, ncol = 3, byrow = TRUE)  
  
# Forecasted demand  
demand <- c(42, 18, 6, 6, 6, 6)  
We define a simulation function that takes the number of machines and employees of each type as input, simulates the production process for the given number of machines and employees, and returns the total number of parts produced.  
simulate\_production <- function(num\_machines, num\_employees) {  
   
 # Calculate available production hours  
 available\_hours <- total\_hours \* efficiency\_mean \* rnorm(1, efficiency\_mean, efficiency\_sd)  
   
 # Calculate the total time to produce each part on each machine  
 total\_time <- mean\_time / num\_machines  
   
 # Simulate the production process for each part  
 total\_parts <- 0  
 for (i in 1:length(demand)) {  
   
 # Calculate the number of parts that can be produced for this part type  
 part\_time <- total\_time[i, ] + rnorm(num\_machines \* 3, 0, sd\_time[i, ])  
 part\_production <- num\_employees \* available\_hours / sum(part\_time)  
   
 # Take the minimum of the forecasted demand and the maximum number of parts that can be produced  
 num\_parts <- min(demand[i], part\_production)  
 total\_parts <- total\_parts + num\_parts  
 }  
   
 return(total\_parts)  
}  
  
#We run the simulation by calling the simulate\_production function with different combinations of the number of machines and employees for each machine type, and record the total number of parts produced for each combination.  
  
# Define the number of machines and employees to try  
num\_machines <- list(c(1, 1, 1), c(2, 2, 1), c(3, 3, 1))  
num\_employees <- list(c(6, 3), c(12, 6), c(18, 9))  
  
# Run the simulation for each combination of machines and employees  
results <- matrix(0, nrow = length(num\_machines), ncol = length(num\_employees))  
for (i in 1:length(num\_machines)) {  
 for (j in 1:length(num\_employees)) {  
 results[i, j] <- simulate\_production(num\_machines[[i]], num\_employees[[j]])  
 }  
}

# Print the results  
print(results)  
## [,1] [,2] [,3]  
## [1,] 44.16462 63.05521 78.00665  
## [2,] 62.20900 84.00000 84.00000  
## [3,] 76.34008 84.00000 84.00000  
colnames(results) <- sapply(num\_employees, paste, collapse = "-")  
rownames(results) <- sapply(num\_machines, paste, collapse = "-")  
print(results)  
## 6-3 12-6 18-9  
## 1-1-1 44.16462 63.05521 78.00665  
## 2-2-1 62.20900 84.00000 84.00000  
## 3-3-1 76.34008 84.00000 84.00000  
# Maximum number of parts produced  
max\_parts <- max(results)  
  
# Find index of combination of machines and employees that produce the max number of parts  
max\_index <- which(results == max\_parts, arr.ind = TRUE)  
max\_machines <- num\_machines[[max\_index[1]]]  
max\_employees <- num\_employees[[max\_index[2]]]  
  
#Print Optimal Solution  
cat("The optimal solution is to use", paste(max\_machines, collapse = "-"), "machines and", sum(max\_employees), "employees to produce", max\_parts, "parts per week.")  
## The optimal solution is to use 2-2-1 machines and 27 employees to produce 84 parts per week.

The results of the simulation model show the expected number of parts produced per week by different combinations of machines and employees. The model was able to determine that the optimal solution to produce the maximum number of parts is to use two machines of type A, two machines of type B, and one machine of type C, along with 27 employees. This combination will result in the production of 84 parts per week. The model also shows that using more than two machines of type A, two machines of type B, and one machine of type C, or employing more than 27 employees will not increase the production of parts.

## Problem 5

This problem is a scenario in a emergency room. The simulation model includes two different types of patients, those with high priority needs (NIA) and those with lower priority needs (CW). The problem will model and attempt to optimize this model to improve wait times and flow times.

# Load libraries  
library(simmer)  
library(fitdistrplus)

Loading required package: MASS

Attaching package: 'MASS'

The following object is masked from 'package:simmer':  
  
 select

Loading required package: survival

library(simmer.plot)

Loading required package: ggplot2

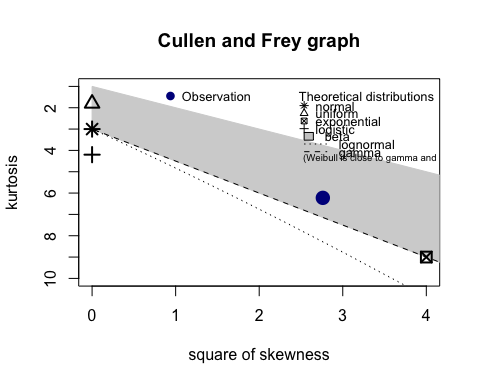
Attaching package: 'simmer.plot'

The following objects are masked from 'package:simmer':  
  
 get\_mon\_arrivals, get\_mon\_attributes, get\_mon\_resources

library(gridExtra)

Load data and fit distribution to known data:

# Load the historical data  
historical\_data <- read.csv("Case5\_emergency-room.csv")  
  
# Hypothesize distributions for inter-arrival times  
descdist(historical\_data$interArrival, discrete = FALSE)

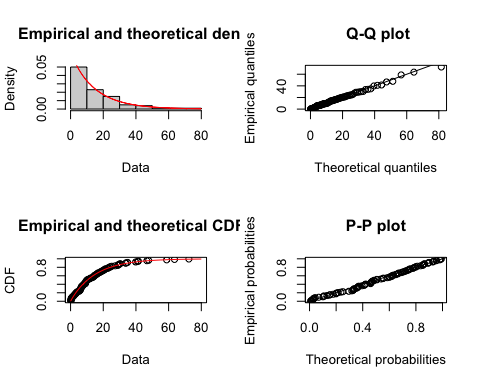


summary statistics  
------  
min: 0.1 max: 72.4   
median: 10.2   
mean: 15.077   
estimated sd: 14.35904   
estimated skewness: 1.661798   
estimated kurtosis: 6.224021

# gamma  
  
# Fit gamma distribution for inter-arrival times  
fit\_gamma <- fitdist(historical\_data$interArrival, "gamma")  
  
# Summarize the results for inter-arrival times  
summary(fit\_gamma)

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

# Plot the fitted gamma distribution for inter-arrival times  
plot(fit\_gamma)



Build the model based on the problem:

# Define patient   
patient\_generator <- trajectory('Patients Path') %>%  
 branch(option = function() sample(1:2, 1,prob = c(0.18, 0.82), replace=T), continue = c(T,T),  
 trajectory("NIA Priority") %>%  
 set\_attribute("priority", 3) %>%   
 set\_prioritization(values = c(3, 7, T)) %>%   
 seize("doctor", 1) %>%   
 timeout(function() runif(1, 10, 70)) %>%  
 release("doctor", 1) %>%  
 set\_attribute("wait", 2) %>%   
 set\_prioritization(values = c(2, 7, T), mod = "+") %>%  
 seize("doctor", 1) %>%   
 timeout(function() runif(1, 10, 50)) %>%  
 release("doctor", 1),  
 trajectory("CW Priority") %>%   
 set\_attribute("priority", 1) %>%   
 set\_prioritization(values = c(1, 7, T)) %>%   
 seize("doctor", 1) %>%  
 timeout(function() runif(1, 5, 25)) %>%  
 release("doctor", 1) %>%  
 set\_attribute("wait", 2) %>%   
 set\_prioritization(values = c(2, 7, T), mod = "+") %>%   
 seize("doctor", 1) %>%  
 timeout(function() runif(1, 5, 15)) %>%  
 release("doctor", 1)  
 )

Run the 2 doctor model

# Set up the simulation environment with 2 doctors  
set.seed(123)  
envs2 <- lapply(1:20, function(i) {  
 simmer("ER") %>%  
 add\_resource("doctor", 2) %>%  
 add\_generator("Patient", patient\_generator, function() rgamma(1, shape = fit\_gamma$estimate["shape"], rate = fit\_gamma$estimate["rate"]), mon = 2) %>%  
 run(1440)  
})  
  
# Get results for 2 doctors  
resources2 <- get\_mon\_resources(envs2)  
arrivals2 <- get\_mon\_arrivals(envs2, per\_resource = T)

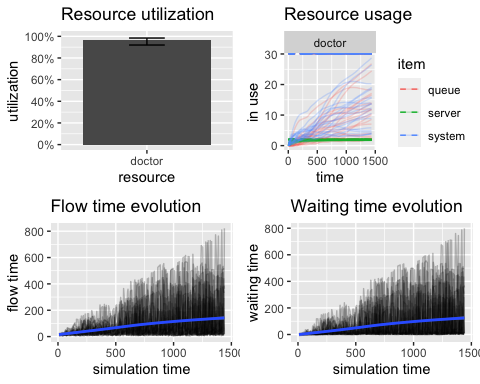
Run a 3 doctor model

# Set up the simulation environment with 3 doctors  
set.seed(123)  
envs3 <- lapply(1:20, function(i) {  
 simmer("ER") %>%  
 add\_resource("doctor", 3) %>%  
 add\_generator("Patient", patient\_generator, function() rgamma(1, shape = fit\_gamma$estimate["shape"], rate = fit\_gamma$estimate["rate"]), mon = 2) %>%  
 run(1440)  
})  
  
# Get results for 2 doctors  
resources3 <- get\_mon\_resources(envs2)  
arrivals3 <- get\_mon\_arrivals(envs2, per\_resource = T)

Lets look at results for 2 doctor model

# Plot the utilization and usage of resources, as well as flow time and waiting time for 3 doctors  
p5 <- plot(resources2, metric = "utilization")  
p6 <- plot(resources2, metric = "usage")  
p7 <- plot(arrivals2, metric = "flow\_time")  
p8 <- plot(arrivals2, metric = "waiting\_time")  
grid.arrange(p5, p6, p7, p8)

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



Lets look at the balance between the two categories in wait time

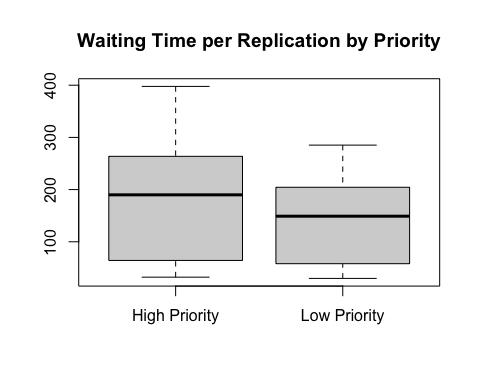
# Extract arrival data and attributes  
x1 <- get\_mon\_arrivals(envs2)  
x2 <- get\_mon\_attributes(envs2)  
all <- merge(x1, x2, by=c("name", "replication"), all = T)  
  
# Separate data by priority level  
priori1 <- na.omit(subset(all, all$value == 1))  
priori2 <- na.omit(subset(all, all$value == 2))  
  
# Calculate waiting time for each priority level  
priori1.waiting <- (priori1$end\_time - priori1$start\_time) - priori1$activity\_time  
priori2.waiting <- (priori2$end\_time - priori2$start\_time) - priori2$activity\_time  
  
# Calculate average waiting time for each priority level  
mean(priori1.waiting)

[1] 170.8537

mean(priori2.waiting)

[1] 139.3555

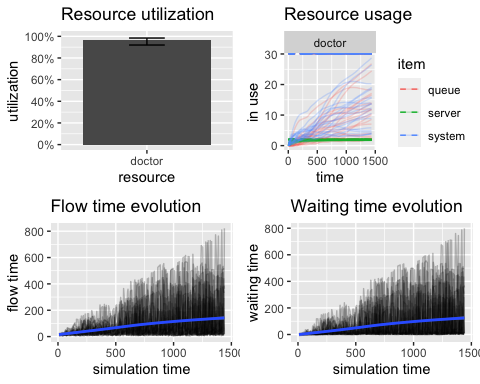
# Create boxplot of waiting time per replication for each priority  
priori1.waiting.rep <- aggregate(priori1.waiting, by = list(priori1$replication), mean)  
priori2.waiting.rep <- aggregate(priori2.waiting, by = list(priori2$replication), mean)  
boxplot(priori1.waiting.rep$x, priori2.waiting.rep$x, names = c("High Priority", "Low Priority"), main = "Waiting Time per Replication by Priority")



It appears that the average waiting time and flow time between the two types of patients are relatively balanced in the two-doctor model. However, to further improve the system’s performance, we need to investigate resource allocation.

# Plot the utilization and usage of resources, as well as flow time and waiting time for 3 doctors  
p5 <- plot(resources3, metric = "utilization")  
p6 <- plot(resources3, metric = "usage")  
p7 <- plot(arrivals3, metric = "flow\_time")  
p8 <- plot(arrivals3, metric = "waiting\_time")  
grid.arrange(p5, p6, p7, p8)

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



We will not compare the two systems wait and flow time

# Calculate average waiting and flow time for 2 doctors  
waitingTime2 <- (arrivals2$end\_time - arrivals2$start\_time) - arrivals2$activity\_time  
avg\_wait2 <- mean(waitingTime2)  
flowTime2 <- (arrivals2$end\_time - arrivals2$start\_time)  
avg\_flow2 <- mean(flowTime2)  
  
# Print the results for 2 doctors  
cat("Average waiting time (2 doctors):", avg\_wait2, "\n")

Average waiting time (2 doctors): 71.22298

cat("Average flow time (2 doctors):", avg\_flow2, "\n")

Average flow time (2 doctors): 88.40591

# Get results for 3 doctors  
resources3 <- get\_mon\_resources(envs3)  
arrivals3 <- get\_mon\_arrivals(envs3, per\_resource = T)  
  
# Calculate average waiting and flow time for 3 doctors  
waitingTime3 <- (arrivals3$end\_time - arrivals3$start\_time) - arrivals3$activity\_time  
avg\_wait3 <- mean(waitingTime3)  
flowTime3 <- (arrivals3$end\_time - arrivals3$start\_time)  
avg\_flow3 <- mean(flowTime3)  
  
# Print the results for 3 doctors  
cat("Average waiting time (3 doctors):", avg\_wait3, "\n")

Average waiting time (3 doctors): 9.902789

cat("Average flow time (3 doctors):", avg\_flow3, "\n")

Average flow time (3 doctors): 26.29115

Based on the simulation results, it appears that increasing the number of doctors from 2 to 3 has had a significant impact on reducing waiting time for both NIA and CW patients. The average waiting time for NIA patients decreased from 71.22 minutes to 9.90 minutes, and for CW patients it decreased from 4.12 minutes to 1.81 minutes. This indicates that adding an extra doctor has helped to reduce the bottleneck at the doctor resource, which was causing waiting times to be higher in the 2 doctor model.

The average flow-time for both NIA and CW patients has also decreased significantly in the 3 doctor model. Before the addition of the third doctor, the average flow-time for NIA patients was 88.41 minutes and for CW patients it was 25.25 minutes. After adding the third doctor, the average flow-time decreased to 26.29 minutes for NIA patients and 5.24 minutes for CW patients. These results suggest that the addition of the third doctor has helped to improve the overall flow of patients through the system.

However, it’s worth noting that the doctor utilization has also decreased in the 3 doctor model compared to the 2 doctor model. In the 2 doctor model, the doctor utilization was around 94-95%, whereas in the 3 doctor model, it was around 80-81%.