HW6

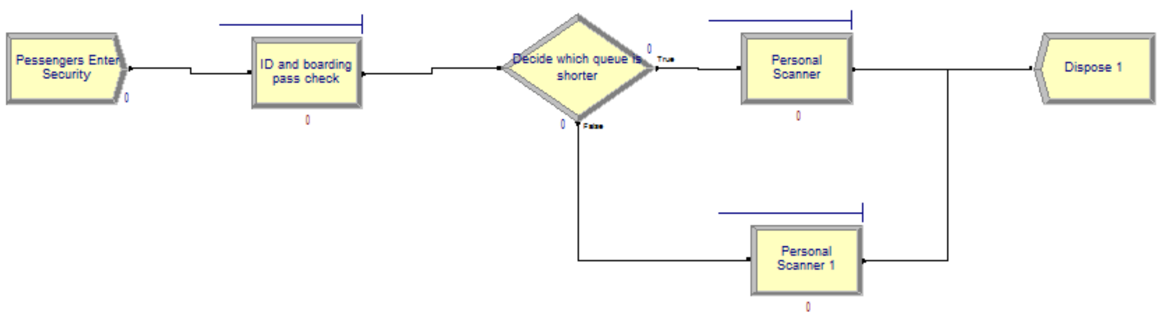
Question 1

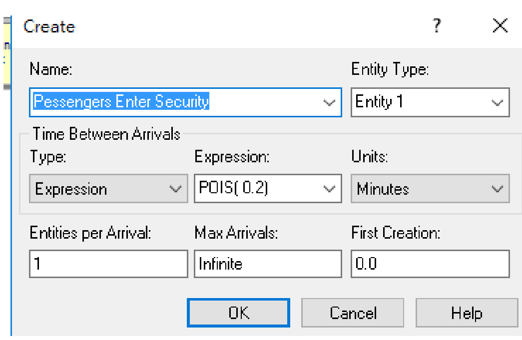
In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with $\lambda_1 = 5$ per minute (i.e., mean interarrival rate $\mu_1 = 0.2$ minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate $\mu_2 = 0.75$ minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute).

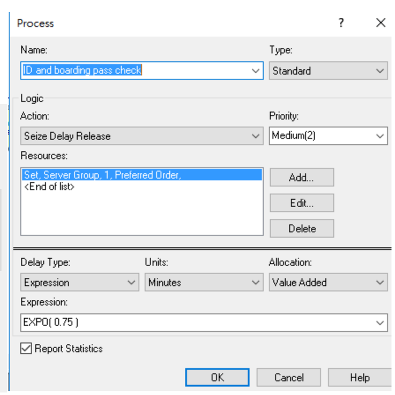
Use the Arena software (PC users) or Python with SimPy (Mac users) to build a simulation of the system, and then vary the number of ID/boarding-pass checkers and personal-check queues to determine how many are needed to keep average wait times below 15 minutes.

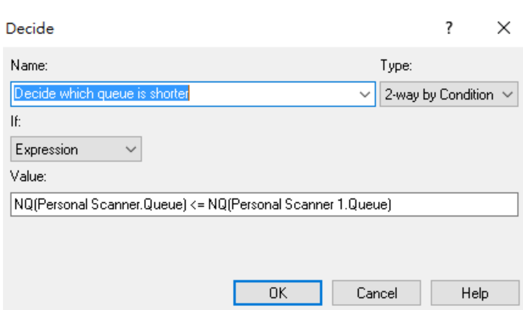
Answer:

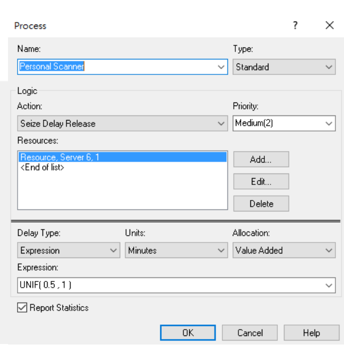
First, try 3 servers at ID & boarding pass check, then 2 personal scanner lines with one server each. Run 10 replications with 5 hours each. The result for avg waiting time is 0.44hr (26.4 min). And the personal scanner line each need to wait about 0.31hr (18.6min), which is already higher than 15 min.







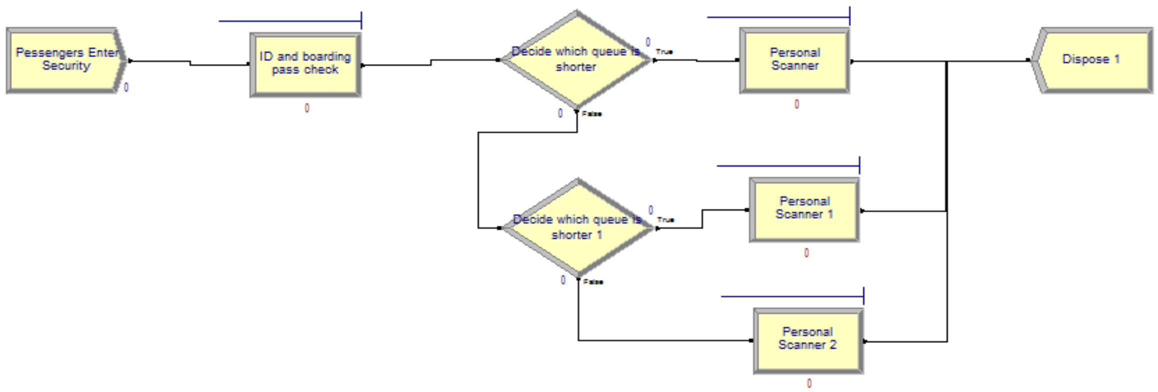




Wait Time	Average
Entity 1	0.4396

Waiting Time	Average
ID and boarding pass check.Queue	0.1300
Personal Scanner 1.Queue	0.3108
Personal Scanner.Queue	0.3114

Second, try to add one personal scanner. The result of avg waiting time is 0.31hr(18.6min). It seems that the waiting time at ID and boarding pass check is too long.



Wait Time	Average
Entity 1	0.3156

Waiting Time	Average
ID and boarding pass check.Queue	0.2636
Personal Scanner 1.Queue	0.05120456
Personal Scanner 2.Queue	0.04611909
Personal Scanner.Queue	0.05924787

Third, try to add additional one server at ID & boarding pass check. The result of avg waiting time is 0.275hr(16.5 min)

Wait Time	Average
Entity 1	0.2756

Waiting Time	Average
ID and boarding pass check.Queue	0.04396864
Personal Scanner 1.Queue	0.2308
Personal Scanner 2.Queue	0.2287
Personal Scanner.Queue	0.2385

Forth, try to add additional one personal scanner. The result of avg waiting time is 0.1088hr (6.528min), which is less than 15 min. Great! Then we use 4 servers at ID check, and use 4 personal scanners.

Wait Time	Average
Entity 1	0.1088

Waiting Time	Average
ID and boarding pass check.Queue	0.07457095
Personal Scanner 1.Queue	0.03570406
Personal Scanner 2.Queue	0.02793915
Personal Scanner 3.Queue	0.02387695
Personal Scanner.Queue	0.04553027

Question 2

The breast cancer data set at http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values.

1. Use the mean/mode imputation method to impute values for the missing data.
2. Use regression to impute values for the missing data.
3. Use regression with perturbation to impute values for the missing data.
4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using (1) the data sets from questions 1,2,3; (2) the data that remains after data points with missing values are removed; and (3) the data set when a binary variable is introduced to indicate missing values.

Answer:

2.1

```
# First, rea the data
data <- read.table("breast-cancer-wisconsin.data.txt", stringsAsFactors = FALSE, header = FALSE, sep=",")

# Try to find the missing value
for (i in 2:11){
   print(paste0("V",i))
   print(table(data[,i]))
}
```

```
[1] "V7"

  ?  1  10   2   3   4   5   6   7   8   9
 16 402 132  30  28  19  30   4   8  21   9
```

⇨ Only V7 with missing 16 values (show in question marks)

```
# show the missing data
data[which(data$V7 =="?"),]
```

```
         V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
24  1057013  8  4  5  1  2  ?  7  3   1   4
41  1096800  6  6  6  9  6  ?  7  8   1   2
140 1183246  1  1  1  1  1  ?  2  1   1   2
146 1184840  1  1  3  1  2  ?  2  1   1   2
159 1193683  1  1  2  1  3  ?  1  1   1   2
165 1197510  5  1  1  1  2  ?  3  1   1   2
236 1241232  3  1  4  1  2  ?  3  1   1   2
250  169356  3  1  1  1  2  ?  3  1   1   2
276  432809  3  1  3  1  2  ?  2  1   1   2
293  563649  8  8  8  1  2  ?  6 10   1   4
295  606140  1  1  1  1  2  ?  2  1   1   2
298   61634  5  4  3  1  2  ?  2  3   1   2
316  704168  4  6  5  6  7  ?  4  9   1   2
322  733639  3  1  1  1  2  ?  3  1   1   2
412 1238464  1  1  1  1  1  ?  2  1   1   2
618 1057067  1  1  1  1  1  ?  1  1   1   2
```

```
# calculate the missing %
nrow(data[which(data$V7 =="?"),])/nrow(data)
[1] 0.02288984
```

=> Smaller than 5%, okay to use imputation. In this case, v7 is not continues number, so should use mode instead of mean to impute. The mode for V7 should be 1. Then use 1 to impute missing values.

```
[1] "V7"

  ?   1  10   2   3   4   5   6   7   8   9
 16 402 132  30  28  19  30   4   8  21   9
```

2.2

Not to include the response variable in regression imputation

data_modified <- data[-missing, 2:10]

data_modified$V7 <- as.integer(data_modified$V7)

run linear regression

model <- lm(V7~V2+V3+V4+V5+V6+V7+V8+V9+V10, data_modified)

summary(model)

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.616652   0.194975  -3.163  0.00163 ** 
V2           0.230156   0.041691   5.521 4.83e-08 ***
V3          -0.067980   0.076170  -0.892  0.37246    
V4           0.340442   0.073420   4.637 4.25e-06 ***
V5           0.339705   0.045919   7.398 4.13e-13 ***
V6           0.090392   0.062541   1.445  0.14883    
V8           0.320577   0.059047   5.429 7.91e-08 ***
V9           0.007293   0.044486   0.164  0.86983    
V10         -0.075230   0.059331  -1.268  0.20524    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.274 on 674 degrees of freedom
Multiple R-squared:  0.615,	Adjusted R-squared:  0.6104 
F-statistic: 134.6 on 8 and 674 DF,  p-value: < 2.2e-16
```

step(model)

```
Call:
lm.default(formula = V7 ~ V2 + V4 + V5 + V8, data = data_modified)

Coefficients:
(Intercept)           V2           V4           V5           V8  
    -0.5360       0.2262       0.3173       0.3323       0.3238  
```

⇨ V2, V4, V5, V8 are important variables to predict V7

model2<- lm(V7~V2+V4+V5+V8, data_modified)

summary(model2)

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.53601    0.17514  -3.060   0.0023 ** 
V2           0.22617    0.04121   5.488 5.75e-08 ***
V4           0.31729    0.05086   6.239 7.76e-10 ***
V5           0.33227    0.04431   7.499 2.03e-13 ***
V8           0.32378    0.05606   5.775 1.17e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '

Residual standard error: 2.274 on 678 degrees of freedom
Multiple R-squared:  0.6129,	Adjusted R-squared:  0.6107 
F-statistic: 268.4 on 4 and 678 DF,  p-value: < 2.2e-16
```

⇨ All variables are significant, use this model to predict V7

predict(impute) V7

V7_hat <- predict(model2, newdata = data[missing,])

```
> V7_hat
       24        41       140       146       159       165       236       250       276 
5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652 1.7634059 2.0741942 
      293       295       298       316       322       412       618 
6.0866099 0.9872832 2.5265324 5.2438347 1.7634059 0.9872832 0.6634986 
```

data_reg_imp <- data

data_reg_imp[missing,]$V7 <-V7_hat

data_reg_imp$V7 <- as.numeric(data_reg_imp$V7)

2.3

V7_hat_pert <- rnorm(length(missing), V7_hat, sd(V7_hat))

V7_hat_pert

```
> V7_hat_pert
 [1]  4.0777220  8.3863924 -0.8545876  5.1381323  1.7070775  0.4072891  3.7896436  3.3908019
 [9]  3.3433164  5.4134808  4.3195118  3.3858147  3.8745124 -3.1181778  3.4668265  0.5644572
```

data_reg_pert_imp <- data

data_reg_pert_imp[missing,]$V7 <- V7_hat_pert

data_reg_pert_imp$V7 <- as.numeric(data_reg_pert_imp$V7)

Question 3

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

Answer:

Use optimization to allocate my time properly to maximize my happiness. (Happiness here is defined in a broad view. For example “rest” is a kind of happiness that can be get by doing activities such as watch TV, sleep or meditation) Following is an optimization model which specify the required data.

n activities

m happiness

aij = amount of happiness j per unit of activity i

mj = minimum daily required of happiness j

Mj = Maximum daily required of happiness j

Variables:

Xi = How many hour spend on activity i daily

Constraints:

Sigma(i)aijXi >= mj for each kind of happiness j (For example, it is must to have nutrition and have rest every day. Therefore set a minimum requirement)

Sigma(i)aijXi <= Mj for each kind of happiness j (For example, rest too much is also not good. Therefore set a Max value)

Sigma(i)Xi <=24 hours (A day only have 24 hours)

Xi >=0

Objective function:

Maximize Sigma(i)Xi * aij