ISYE6501 Homework week 6

Dylan Peters

June 21, 2017

Question 1

Arena Simluation:

I created a simulation with 4 ID checker resources as part of an ID checker set and a single queue. The next process was determined by a decision to figure out the shortest of 4 personal scanner queues. The simulation was run for 12 hours with 7 repetitions–I encountered the 150 entity limit in Arena with this configuration. However the average total wait time was 0.1445 hours or 8.67 minutes, well below the 15 minutes threshold. I could not lower the number of queues without quickly hitting the 150 entity limit.

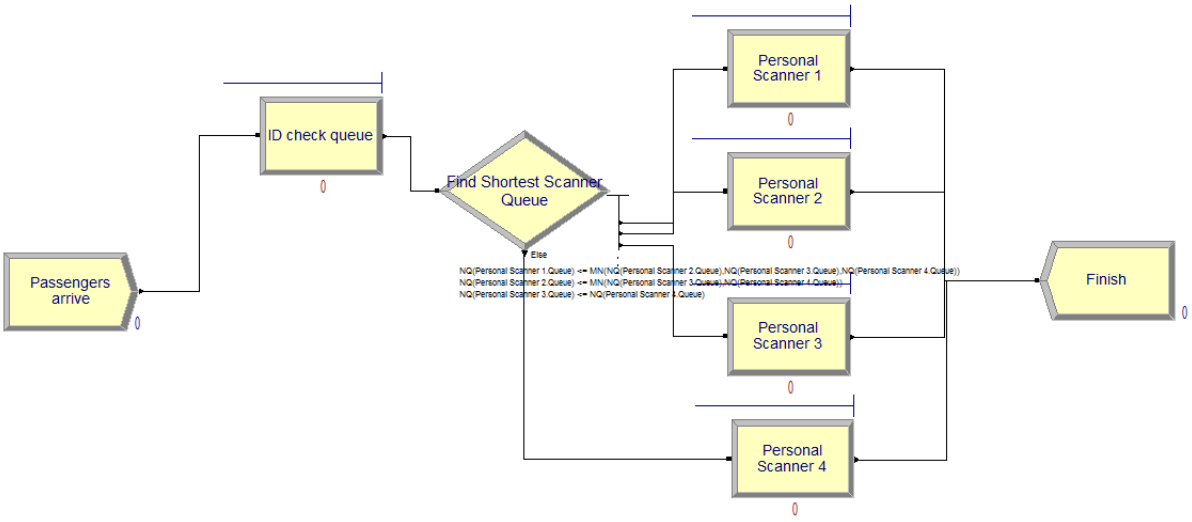


Figure 1: alt text

I increased the number of ID checkers in the queue from 4 to 5, however I still hit the 150 limit because the 4 personal scanners filled up. After adding one more personal scanner queue, I let the simulation run for 100 repetitions. The final average total time was 0.0589 hours or 3.534 minutes.

The conclusion is that 4 ID checkers and 4 scanners are a feasible solution. It is likely the optimal solution. Using only 3 ID checkers leads to a generally increasing queue and thus the wait times continue to grow.

Question 2

Cleaning missing data:

First load the data:

```
# load the data and reset the target variable
# Make sure that the ? values are interpreted as NA
raw.data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
# (2 for benign, 4 for malignant)
raw.data$v11[raw.data$V11 == 2] <- 0
raw.data$v11[raw.data$V11 == 4] <- 1

raw.data$v11 <- as.factor(raw.data$v11)
```

1. Use the mean/mode imputation method to impute values for the missing data.

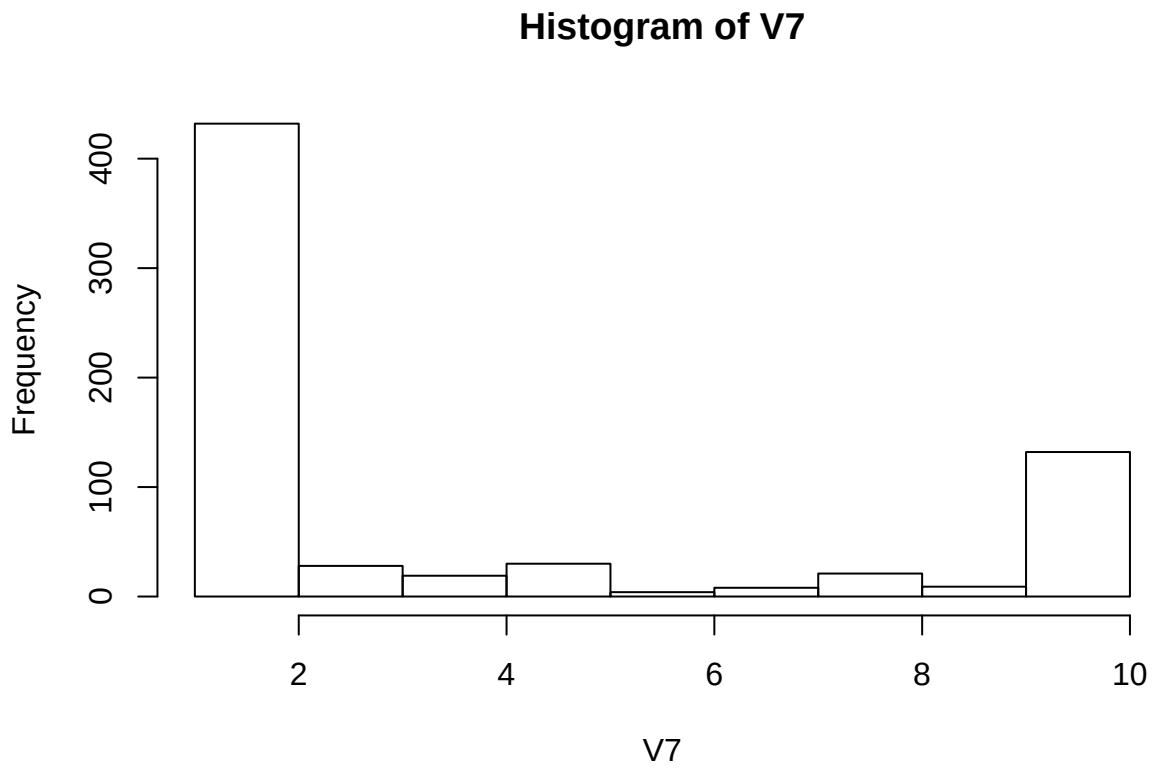
```
# Analyze the data.

sum(is.na(raw.data$V7))
```

```
## [1] 16
```

```
# 16 NA values in V7
# Get the non-NA values
V7 <- raw.data$V7[!is.na(raw.data$V7)]

hist(V7)
```



The values are mostly 1, some 10, and a few in between. A choice needs to be made whether to calculate the mean of the variable, or use the mode of 1. I am choosing to use the mode for several reasons. The data is not normally distributed; the values are all integers so don't appear continuous; and there are no values outside the range 1-10.

```
data.mode <- raw.data
data.mode$V7[is.na(data.mode$V7)] <- 1
```

2. Use regression to impute values for the missing data.

Be sure not to include the row name (V1) or the response variable (V11):

```
data.reg <- raw.data[!is.na(raw.data$V7),-c(1,11)]

reg.model <- lm(V7 ~ ., data=data.reg)
predicted <- predict(reg.model, raw.data[is.na(raw.data$V7),])
data.regressed <- raw.data
data.regressed$V7[is.na(data.regressed$V7)] <- predicted
```

3. Use regression with perturbation to impute values for the missing data.

We can use the same model as the previous setup, just add some variability.

```
data.perturbed <- raw.data
# get the prediction intervals and create perturbations from that
predicted.interval <- predict(reg.model, raw.data[is.na(raw.data$V7),],
                              interval="predict", se.fit=TRUE)
data.perturbed$V7[is.na(data.perturbed$V7)] <-
  rnorm(data.perturbed$V7[is.na(data.perturbed$V7)],
        mean=predicted.interval$fit[,1],
        sd=predicted.interval$se.fit)
```

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.

Create a cleaned dataset and an indicator dataset.

```
data.cleaned <- raw.data[!is.na(raw.data$V7),]

data.indicator <- raw.data
data.indicator$V7Missing <- as.integer(is.na(data.indicator$V7))
data.indicator$V7[is.na(data.indicator$V7)] <- 0
```

Create the models

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
kfolds <- createFolds(raw.data$V11)
train_control <- trainControl(method="cv", number=10, indexOut=kfolds)

model.mode <- train(V11 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
                    data=data.mode,
                    trControl = train_control,
                    method="svmLinear")
```

```
## Loading required package: kernlab
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
## do classification? If so, use a 2 level factor as your outcome column.
```

```
model.mode$results
```

```
##   C      RMSE  Rsquared     RMSESD RsquaredSD
## 1 1 0.3882249 0.8332348 0.04262449 0.03748979
```

```
model.regressed <- train(V11 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
                 data=data.regressed,
                 trControl = train_control,
                 method="svmLinear")
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
## do classification? If so, use a 2 level factor as your outcome column.
```

```
model.regressed$results
```

```
##   C      RMSE  Rsquared     RMSESD RsquaredSD
## 1 1 0.3866371 0.8344406 0.04097905 0.03514066
```

```
model.perturbed <- train(V11 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
                 data=data.perturbed,
                 trControl = train_control,
                 method="svmLinear")
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
## do classification? If so, use a 2 level factor as your outcome column.
```

```
model.perturbed$results
```

```
##   C      RMSE Rsquared     RMSESD RsquaredSD
## 1 1 0.3871798 0.834277 0.03934251 0.03338377
```

```
model.indicator <- train(V11 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
                 data=data.indicator,
                 trControl = train_control,
                 method="svmLinear")
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
## do classification? If so, use a 2 level factor as your outcome column.
```

```
model.indicator$results
```

```
##   C      RMSE  Rsquared     RMSESD RsquaredSD
## 1 1 0.3899118 0.8315195 0.04372634 0.03822936
```

```
kfolds2 <- createFolds(data.cleaned$V11)
train_control2 <- trainControl(method="cv", number=10, indexOut=kfolds2)
```

```
model.cleaned <- train(V11 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
                  data=data.cleaned,
                  trControl = train_control2,
                  method="svmLinear")
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
## do classification? If so, use a 2 level factor as your outcome column.
```

```
model.cleaned$results
```

```
##   C      RMSE  Rsquared     RMSESD RsquaredSD
## 1 1 0.3797877 0.8423094 0.07202571 0.05071849
```

In this scenario, the cleaned dataset provided the best performance as measured by RMSE and R squared.

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?

The software I work on helps analyze audio data for call centers in order to maintain compliance and evaluate call agents. This includes workflows where calls are routed to different queues based on call characteristics such as customer complaints, retention, etc. Large customers can process thousands of calls per day, so the analytics software generates hundreds of calls for the evaluators to verify. Calls that go to one queue can be processed by one of several evaluators, and may be escalated to a higher priority or switched to a legal or compliance queue. A simulation could model how many evaluators are needed to process each queue. The import process could be modeled with a specific rate of calls for each input queue, a process time for each queue, and a random assignment of calls to sub-queues. This could model throughput and help find an optimal solution that will reduce processing time, lower costs, and ensure the business stays in legal compliance.