Week 6 HW

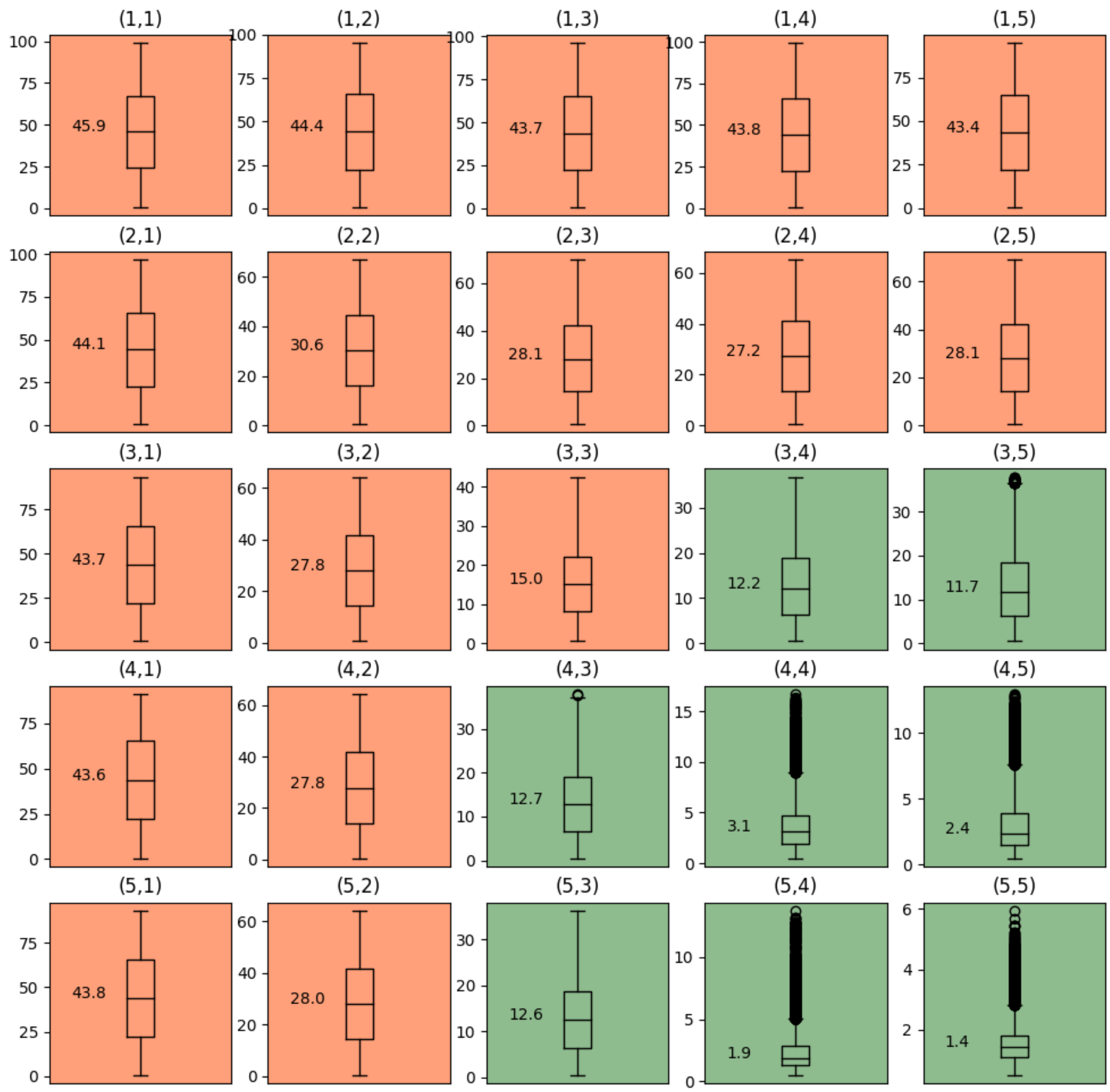
Question 13.2

Simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute (i.e., mean interarrival rate μ1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate μ2 = 0.75 minutes. After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner whose time is drawn from a uniform distribution between 0.5 minutes and 1 minute

Use `SimPy` 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.

For this problem, I built a simulation model in `SimPy` that is documented in the file `Airport.py`. I ran 100 trials to get average wait times over 120 minutes for each combination of 1 through 5 checkers and 1 through 5 scanners, and got wait times shown in the box plots below. The row represents the number of checkers and the column represents the number of scanners. The median is denoted in each boxplot, and the configurations with median wait times below 15 minutes are colored in green.

Airport wait times



Question 14.1

The breast cancer data set `breast-cancer-wisconsin.data.txt` *has missing values.*

Prompt 1:

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

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

# Find where the missing data is
for (i in 2:11) {
  print(paste0("V",i))
  print(table(data[,i]))
}
```

```
## [1] "V2"
##
##   1   2   3   4   5   6   7   8   9  10
## 145  50 108  80 130  34  23  46  14  69
## [1] "V3"
##
##   1   2   3   4   5   6   7   8   9  10
## 384  45  52  40  30  27  19  29   6  67
## [1] "V4"
##
##   1   2   3   4   5   6   7   8   9  10
## 353  59  56  44  34  30  30  28   7  58
## [1] "V5"
##
##   1   2   3   4   5   6   7   8   9  10
## 407  58  58  33  23  22  13  25   5  55
## [1] "V6"
##
##   1   2   3   4   5   6   7   8   9  10
##  47 386  72  48  39  41  12  21   2  31
## [1] "V7"
##
##   ?   1  10   2   3   4   5   6   7   8   9
##  16 402 132  30  28  19  30   4   8  21   9
## [1] "V8"
##
##   1   2   3   4   5   6   7   8   9  10
## 152 166 165  40  34  10  73  28  11  20
## [1] "V9"
##
##   1   2   3   4   5   6   7   8   9  10
## 443  36  44  18  19  22  16  24  16  61
## [1] "V10"
##
##   1   2   3   4   5   6   7   8  10
## 579  35  33  12   6   3   9   8  14
## [1] "V11"
##
##   2   4
## 458 241
```

We see that V7 is the only column with missing data. Let’s try mean/mode imputation on the missing values.

```
missing <- which(data$V7 == "?", arr.ind = TRUE)

# Convert V7 to a numeric datatype
data$V7 <- as.integer(data$V7)

getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}


# Find the mode of V7
mode_V7 <- as.numeric(getmode(data[-missing,"V7"]))
mode_V7
```

```
## [1] 2
```

```
# Find the mean of V7
mean_V7 <- mean(as.numeric(data[-missing, "V7"]))
mean_V7
```

```
## [1] 3.216691
```

```
# Impute V7 for observations with missing data for V7 to mode_v7
data_mode_imp <- data
data_mode_imp[missing,]$V7 <- mode_V7
data_mode_imp$V7 <- as.integer(data_mode_imp$V7)

# Impute V7 for observations with missing data for V7 to mean_v7
data_mean_imp <- data
data_mean_imp$V7 <- as.integer(data_mean_imp$V7)
data_mean_imp[missing,]$V7 <- mean_V7
data_mean_imp$V7 <- as.integer(data_mean_imp$V7)
```

Prompt 2:

Use regression to mipute the values for the missing data.

```
data_modified <- data[-missing,2:10]
data_modified$V7 <- as.integer(data_modified$V7)

# Generate a linear model using all other factors as predictors for V7
model <- lm(V7~., data = data_modified)
summary(model)
```

```
##
## Call:
## lm(formula = V7 ~ ., data = data_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1137 -0.7185 -0.4731 -0.2994  7.3848
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.862817   0.162497  11.464  < 2e-16 ***
## V2           0.068118   0.034746   1.960  0.05035 .
## V3           0.087939   0.063482   1.385  0.16643
## V4           0.110046   0.061190   1.798  0.07255 .
## V5          -0.076950   0.038270  -2.011  0.04475 *
## V6           0.043216   0.052123   0.829  0.40733
## V8           0.044536   0.049211   0.905  0.36579
## V9           0.119422   0.037076   3.221  0.00134 **
## V10          0.001405   0.049448   0.028  0.97733
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.896 on 674 degrees of freedom
## Multiple R-squared:  0.2326, Adjusted R-squared:  0.2235
## F-statistic: 25.54 on 8 and 674 DF,  p-value: < 2.2e-16
```

```
# Get predictions for missing V7 values
V7_hat <- predict(model, newdata = data[missing,])
V7_hat
```

```
##       24       41      140      146      159      165      236      250
## 3.990654 4.294718 2.305086 2.568395 2.457029 2.665311 2.859213 2.529074
##      276      293      295      298      316      322      412      618
## 2.704631 5.463968 2.348302 3.343528 4.308317 2.529074 2.305086 2.260550
```

```
# Impute V7 for observations with missing data for V7
data_reg_imp <- data
data_reg_imp$V7 <- as.numeric(data_reg_imp$V7)
data_reg_imp[missing,]$V7 <- V7_hat

# Round the V7_hat values since the originals are all integer
data_reg_imp[missing,]$V7 <- round(V7_hat)
data_reg_imp$V7 <- as.integer(data_reg_imp$V7)

# Make sure no V7 values are outside of the original range
data_reg_imp$V7[data_reg_imp$V7 > 10] <- 10
data_reg_imp$V7[data_reg_imp$V7 < 1] <- 1
```

Prompt 3:

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

```
# Perturb the V7_hat values generated in the previous step, drawing from a random nor
mal distribution with standard deviation equal to that of V7_hat.
V7_hat_pert <- rnorm(nrow(data[missing,]), V7_hat, sd(V7_hat))
V7_hat_pert
```

```
##  [1] 4.106661 2.476565 2.649268 3.916368 4.521800 3.310974 2.785377
##  [8] 2.800360 2.596081 6.015489 1.485695 3.934190 4.985861 1.535584
## [15] 1.092398 1.463880
```

```
# Apply the perturbed imputed V7 values for observations with missing V7 values
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)

# Round the V7_hat_pert values to integers
data_reg_pert_imp[missing,]$V7 <- round(V7_hat_pert)
data_reg_pert_imp$V7 <- as.integer(data_reg_pert_imp$V7)

# Make sure no V7 values are outside of the original range
data_reg_pert_imp$V7[data_reg_pert_imp$V7 > 10] <- 10
data_reg_pert_imp$V7[data_reg_pert_imp$V7 < 1] <- 1
```

Question 15.1

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

A bitcoin miner may want to optimize their profit from mining bitcoins. They would need data on the cost of hardware, the power consumption for different components, the price of power, the failure rate of hardware, and the projected hash rate for different configurations. They would also need a model for the future price of bitcoin, and the future network hash rate.

Some constraints in the system would be that only certain hardware components can go together, and that none of the quantities involved can be negative. It's probably reasonable to assume that the network hash rate will go up over time, although that probably depends on the future price of bitcoin. The miner may wish to create a simulation so they can test their configuration for different models that predict the future price of bitcoin.