# A Site Selection and Recruitment Strategy of JPTA Program Evaluation: Data Analysis

DS4SI Assignment 2

*Tong Jin*
*Steinhardt A3SR*

*09/30/2019*

```r
# Import the JPTA dataset
jpta <- read.csv("jpta.csv", header = TRUE)
N <- dim(jpta)[1]
## Step 1:
# Create 4 subsets and group the dataset by regions
jpta_1 <- subset(jpta, jpta$region == 1) # Northeast
jpta_2 <- subset(jpta, jpta$region == 2) # North Central
jpta_3 <- subset(jpta, jpta$region == 3) # South
jpta_4 <- subset(jpta, jpta$region == 4) # West

# Subset by urban (1 for urban area and 0 for rural area)
# Sort the comfort variable in descending order
jpta_1_1 <- jpta_1 %>% # Northeast Urban
              subset(jpta_1$urban == 1) %>%
              arrange(desc(comfort))

jpta_1_0 <- jpta_1 %>% # Northeast Rural
              subset(jpta_1$urban == 0) %>%
              arrange(desc(comfort))

jpta_2_1 <- jpta_2 %>% # North Central Urban
              subset(jpta_2$urban == 1) %>%
              arrange(desc(comfort))

jpta_2_0 <- jpta_2 %>% # North Central Rural
              subset(jpta_2$urban == 0) %>%
              arrange(desc(comfort))

jpta_3_1 <- jpta_3 %>% # South Urban
              subset(jpta_3$urban == 1) %>%
              arrange(desc(comfort))

jpta_3_0 <- jpta_3 %>% # South Rural
              subset(jpta_3$urban == 0) %>%
              arrange(desc(comfort))

jpta_4_1 <- jpta_4 %>% # West Urban
              subset(jpta_4$urban == 1) %>%
              arrange(desc(comfort))

jpta_4_0 <- jpta_4 %>% # West Rural
              subset(jpta_4$urban == 0) %>%
```

```r
              arrange(desc(comfort))
```

```r
## Step 2: Determine the weight of each subset
w_1_0 <- dim(jpta_1_0)[1]/N
w_1_1 <- dim(jpta_1_1)[1]/N
w_2_0 <- dim(jpta_2_0)[1]/N
w_2_1 <- dim(jpta_2_1)[1]/N
w_3_0 <- dim(jpta_3_0)[1]/N
w_3_1 <- dim(jpta_3_1)[1]/N
w_4_0 <- dim(jpta_4_0)[1]/N
w_4_1 <- dim(jpta_4_1)[1]/N
```

```r
## Step 3: Create variables with empty vectors to store results
total_comfort <- rep(0, times = N)  # The sum of comfort
per_comfort <- rep(0, times = N)  # The average comfort
total_cost <- rep(0, times = N)  # The sum of cost

# Calculate the sum of comfort, the average comfort and the sum of cost when
# n sites are selected (1 <= n <= 400)
for (n in 1:400) {
    comfort_1_0 <- sum(jpta_1_0[1:round(n * w_1_0), ]$comfort)
    comfort_1_1 <- sum(jpta_1_1[1:round(n * w_1_1), ]$comfort)
    comfort_2_0 <- sum(jpta_2_0[1:round(n * w_2_0), ]$comfort)
    comfort_2_1 <- sum(jpta_2_1[1:round(n * w_2_1), ]$comfort)
    comfort_3_0 <- sum(jpta_3_0[1:round(n * w_3_0), ]$comfort)
    comfort_3_1 <- sum(jpta_3_1[1:round(n * w_3_1), ]$comfort)
    comfort_4_0 <- sum(jpta_4_0[1:round(n * w_4_0), ]$comfort)
    comfort_4_1 <- sum(jpta_4_1[1:round(n * w_4_1), ]$comfort)

    total_comfort[n] <- sum(comfort_1_0, comfort_1_1, comfort_2_0, comfort_2_1,
        comfort_3_0, comfort_3_1, comfort_4_0, comfort_4_1)

    per_comfort[n] <- total_comfort[n]/n

    cost_1_0 <- sum((jpta_1_0[1:round(n * w_1_0), ]$distance) * 500)
    cost_1_1 <- sum((jpta_1_1[1:round(n * w_1_1), ]$distance) * 500)
    cost_2_0 <- sum((jpta_2_0[1:round(n * w_2_0), ]$distance) * 500)
    cost_2_1 <- sum((jpta_2_1[1:round(n * w_2_1), ]$distance) * 500)
    cost_3_0 <- sum((jpta_3_0[1:round(n * w_3_0), ]$distance) * 500)
    cost_3_1 <- sum((jpta_3_1[1:round(n * w_3_1), ]$distance) * 500)
    cost_4_0 <- sum((jpta_4_0[1:round(n * w_4_0), ]$distance) * 500)
    cost_4_1 <- sum((jpta_4_1[1:round(n * w_4_1), ]$distance) * 500)

    total_cost[n] <- sum(cost_1_0, cost_1_1, cost_2_0, cost_2_1, cost_3_0, cost_3_1,
        cost_4_0, cost_4_1) + 1000 * n
}
```

```r
## Step 4: Determine at least how many sites should we select so that the sum
## of comfort can reach 30
N_30 <- 400 - sum(total_comfort >= 30) + 1
# Determine the max of the average comfort from 41 to 400
N_ave_max <- N_30 - 1 + which.max(per_comfort[41:400])
```

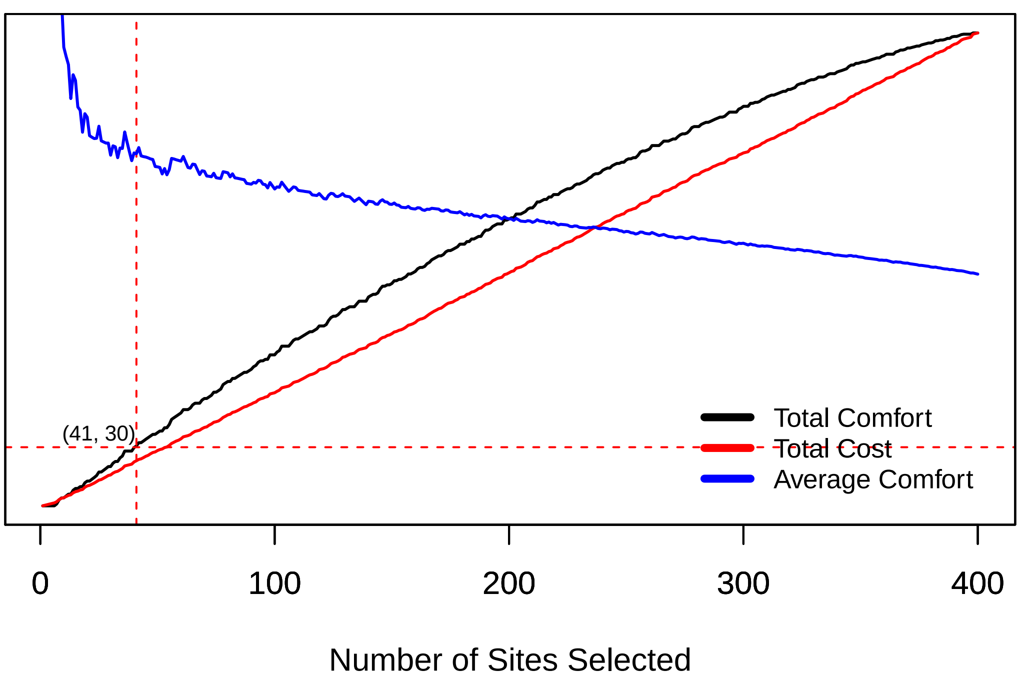
```r
## Step 5: Creates plots to visualize the result Total comfort
plot(total_comfort, type = "l", lty = 1, lwd = 1.5, col = "black", ann = FALSE,
```

```
    yaxt = "n")
# Add line to show when the sum of comfort reaches 30
abline(h = 30, lty = 2, lwd = 1, col = 2)
# Add line to show when the number of sites selected is equal to 41
abline(v = 41, lty = 2, lwd = 1, col = 2)
text(x = 25, y = 35, "(41, 30)", cex = 0.7)

par(new = TRUE)
# Total cost
plot(total_cost, type = "l", lty = 1, lwd = 1.5, col = "red", ann = FALSE, yaxt = "n")

par(new = TRUE)
# Average comfort
plot(per_comfort, type = "l", lty = 1, lwd = 1.5, col = "blue", ann = FALSE,
    yaxt = "n", ylim = c(0, 1))
title(main = "The total comfort, average comfort and total cost when selecting N sites",
    xlab = "Number of Sites Selected", cex.main = 0.9)
legend("bottomright", inset = 0.03, legend = c("Total Comfort", "Total Cost",
    "Average Comfort"), col = c("black", "red", "blue"), lty = c(1, 1, 1), lwd = c(4,
    4, 4), cex = 0.8, box.lty = 0)
```

**The total comfort, average comfort and total cost when selecting N sites**




```
## Step 6: Determine which 41 sites should we select
N <- N_30
sample_1_0 <- jpta_1_0[1:round(N * w_1_0), ]
sample_1_1 <- jpta_1_1[1:round(N * w_1_1), ]
sample_2_0 <- jpta_2_0[1:round(N * w_2_0), ]
sample_2_1 <- jpta_2_1[1:round(N * w_2_1), ]
sample_3_0 <- jpta_3_0[1:round(N * w_3_0), ]
sample_3_1 <- jpta_3_1[1:round(N * w_3_1), ]
```

```r
sample_4_0 <- jpta_4_0[1:round(N * w_4_0), ]
sample_4_1 <- jpta_4_1[1:round(N * w_4_1), ]

sample <- rbind(sample_1_0, sample_1_1, sample_2_0, sample_2_1, sample_3_0,
    sample_3_1, sample_4_0, sample_4_1) %>% arrange(site_id)

# Calculate the total cost
Cost <- 1000 * N + sum(sample$distance) * 500
```

```r
# The total number of sites is:
dim(sample)[1]
```

```
## [1] 41
```

```r
# The total cost of selecting 41 sites is:
paste("$", round(Cost, digits = 2))
```

```
## [1] "$ 84337.63"
```

```r
# The list of sample sites:
data.frame(sample$site_id, sample$region, sample$distance, sample$comfort)
```

```
##    sample.site_id sample.region sample.distance sample.comfort
## 1            1085             4       2.5534772      0.6793509
## 2            1374             2       2.2719130      0.7871178
## 3            1802             3       2.7949002      0.7906509
## 4            1895             2       2.7248187      0.6823765
## 5            2082             4       2.0700336      0.6944425
## 6            2096             4       0.9828134      0.7505317
## 7            2708             1       0.8369433      0.7144757
## 8            2816             1       1.3842986      0.7654412
## 9            2901             3       2.7871634      0.7672680
## 10           2951             1       1.8131748      0.7123354
## 11           3075             2       0.6239017      0.6793425
## 12           3232             1       2.1004726      0.9328601
## 13           3396             4       1.0863968      0.7427873
## 14           3860             4       1.8536565      0.7121777
## 15           4118             1       1.9800302      0.6688466
## 16           4429             3       2.8258375      0.8211312
## 17           4540             1       0.8029432      0.7601828
## 18           4807             4       1.9432895      0.7542857
## 19           5091             4       2.7301329      0.6775037
## 20           5095             4       2.3858727      0.7045969
## 21           5263             1       1.5362527      0.7261004
## 22           5336             4       1.9907189      0.7801155
## 23           5454             3       2.3391063      0.7361615
## 24           5514             2       2.2454205      0.8088736
## 25           5545             2       2.2046690      0.6606413
## 26           5767             1       1.9574461      0.6942057
## 27           5907             2       1.4495354      0.6498019
## 28           5948             3       2.8606709      0.7119592
## 29           5977             1       2.3926313      0.7056639
## 30           6533             3       3.7435490      0.8374992
## 31           6759             1       1.9093724      0.7440476
## 32           7154             4       1.9727270      0.9197648
## 33           7440             3       3.3886186      0.8198539
```

```
## 34            8164              3        3.1234164       0.9402033
## 35            8648              1        2.4795256       0.6859688
## 36            8744              4        2.0059626       0.7154524
## 37            9446              2        1.7756025       0.7450564
## 38            9582              3        1.8655949       0.7176159
## 39            9670              3        2.9613396       0.7220163
## 40            9681              2        2.0548630       0.6714709
## 41            9714              4        1.8661634       0.7314373
```