# HW3

Hayden Atchley

September 28, 2021

## 3.1

In this problem we are developing a gravity trip distribution model of the form

$$T_{ij} = \frac{P_i A_j^* (t_{ij})^{-b}}{\sum_{j' \in J} A_{j'}^* (t_{ij'})^{-b}}.$$

Much of the code required to do so was provided in the problem statement; it is provided below:

```r
#' Gravity Model
#' @param p vector of productions, length n
#' @param A vector of attractions, lenth n
#' @param C matrix of impedances, dim n x n
#' @param b impedance parameter
gravity <- function(p, a, C, b){
  # output matrix (all 0 here)
  trips <- matrix(0, nrow = length(p), ncol = length(a))
  # loop over all rows (production)
  for (i in 1:length(p)) {
    bottomA <- sum(a * C[i, ]^(-b)) # denominator

    # loop over all columns (attraction)
    for (j in 1:length(a)) {
      # calculate gravity model for trips from i to j
      topA <- a[j] * C[i,j]^(-b)
      trips[i, j] <-  p[i] * topA / bottomA
    }
  }

  return(trips)
}
```

```r
#' Function to balance gravity model
#' @param p vector of productions, length n
#' @param A vector of attractions, length n
#' @param C matrix of impedances, dim n x n
#' @param b impedance parameter
#' @param tolerance Acceptable change in trips matrix
balance_gravity <- function(p, a, C, b, tolerance) {

  # define starting values
  k <- 0 #iteration counter
  astar <- a # starting unadjusted attractions
  trips0 <- matrix(0, nrow = length(p), ncol = length(a)) #initial T is 0's
  error <- Inf # first time through, error is Infinite

  # loop through algorithm
  while(error > tolerance){
    # compute gravity model with adjusted attractions, using your function
    trips <- gravity(p, astar, C, b)

    # calculate the error as the change in trips in successive iterations
    error <- sum(abs(trips - trips0))

    # protect against infinite loops, increment values
    if (k > 100) break # maximum of 100 iterations
    k <- k + 1
    trips0 <- trips
    astar <- astar * a / colSums(trips) # next iteration astar
  }

  return(trips)
}
```

We also were given a 3-zone system with its respective production/attraction rates and costs:

```r
prod <- c(100, 200, 100)
attr <- c(200, 50, 150)
costs <- matrix(c(2, 5, 4,
                  5, 2, 3,
                  4, 3, 2),
                byrow = T,
                nrow = 3)
```

Using this balanced gravity model and the provided system, we can calculate the trip distribution:

```
dist1 <- balance_gravity(a = attr, p = prod, C = costs, b = 0.5, tolerance = 0.01)
dist1
```

```
##          [,1]      [,2]     [,3]
## [1,] 62.50975  8.329009 29.16124
## [2,] 91.54031 30.492846 77.96684
## [3,] 45.94993 11.178146 42.87193
```

If we check the row and column sums, we find that they match our production and attraction vectors:

```
rowSums(dist1)
```

```
## [1] 100 200 100
```

```
colSums(dist1)
```

```
## [1] 200  50 150
```

## 3.2

Now we need to calibrate the model to match the observed trip matrix:

Observed Trips

| i -> j | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 80 | 5 | 15 |
| 2 | 80 | 40 | 80 |
| 3 | 40 | 5 | 55 |

The only variable in the model that can be adjusted is $b$, the "impedance parameter", i.e. how much the trip cost affects trip distribution. I decided to use root mean squared error as the measure of how well the model matches, and wrote the following code to find the optimal value of $b$:

```
obs <- matrix(c(80, 5, 15,
              80, 40, 80,
              40, 5, 55),
              byrow = T,
              nrow = 3)
optB <- function(b){
  (balance_gravity(prod, attr, costs, b, tolerance = 0.01) - obs)^2 %>%
    sum() %>%
    "*"(1/(nrow(obs)*ncol(obs))) %>%
    sqrt()
}

b <- optimize(optB, lower = -1, upper = 5)$minimum
b
```

```
## [1] 1.404853
```

Now to use that value in the model:

```
balance_gravity(prod, attr, costs, b, tolerance = 0.01) %>%
  round(digits = 1) %>%
  as_tibble() %>%
  add_column(c(1,2,3), .before = 1) %>%
  `colnames<-`(c("i -> j", "1", "2", "3")) %>%
  my_flextable() %>%
  vline(j = 1) %>%
  add_header_lines("Calibrated Model Trip Distribution")
```

Calibrated Model Trip Distribution

| i -> j | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 81.4 | 3.1 | 15.5 |
| 2 | 79.2 | 38.9 | 81.9 |
| 3 | 39.4 | 8.0 | 52.6 |