Spatial microsimulation in R: a beginner's guide to iterative proportional fitting (IPF)

==========

Introduction

This document demonstrates how iterative proportional fitting (IPF) can be performed in R for spatial microsimulation. The aim is to selectively sample individuals from a survey dataset to fill areas about which only aggregated values are known. This procedure provides the context for a paper entitled "'Truncate, replicate, sample': a method for creating integer weights for spatial microsimulation".

If you are interested in developing IPF-related algorithms in R, an updated version of the code is available at the following github repository: https://github.com/Robinlovelace/IPF-performance-testing . The aim is to make the IPF code more generalisable, faster and more compact. Commits/forks welcome.

Input data

Spatial microsimulation requires two input datasets: individual-level data, where rows represent individuals, and geographically aggregated data, where rows represent areas. The following code creates example datasets, based on a survey of 10 individuals and 5 small areas. The spatial microsimulation model will select individuals based on age, sex and mode of transport. For consistency with the (larger) model used for the paper, we will refer to the individual-level data as USd (short for Understanding Society dataset) and the geographic data as all.msim (all constraint variables).

Individual-level data

```
# Read in the data in long form (normaly read.table() used)
c.names <- c("id", "age", "sex", "mode")
USd <- c(       1, 27, "m", "car.d",
                2, 54, "m", "car.d",
                3, 35, "f", "bus",

                4, 42, "m", "walk",
                5, 50, "f", "car.p",
                6, 19, "m", "car.d",

                7, 62, "f", "car.d",
```

```
                8, 21, "f", "bicycle",
                9, 38, "f", "walk",
                10, 48, "f", "car.d")
USd <- matrix(USd, nrow = 10, byrow = T) # Convert long data into matrix, by row
USd <- data.frame(USd) # Convert this into a dataframe
names(USd) <- c.names # Add correct column names
USd$age <- as.numeric(levels(USd$age)[USd$age]) # Age is a numeric variable

USd # Show the data frame in R
```

```
##    id age sex    mode
## 1   1  27   m   car.d
## 2   2  54   m   car.d
## 3   3  35   f     bus
## 4   4  42   m    walk
## 5   5  50   f   car.p
## 6   6  19   m   car.d
## 7   7  62   f   car.d
## 8   8  21   f bicycle
## 9   9  38   f    walk
## 10 10  48   f   car.d
```

Geographical data

```
# Read in the data in long form (normaly read.table() used)
category.labels <- c("16-30", "31-50", "50+" # Age constraint
             ,"m", "f" # Sex constraint
             ,"bicycle", "bus", "car.d", "car.p", "walk" # Mode constraint
             )
all.msim <- c(  3, 3, 4,   5, 5,   0.001, 1, 8, 1, 0.001, # Car dominated
                2, 2, 6,   4, 6,   0.001, 3, 5, 1, 1, # Elderly
                3, 4, 4,   3, 8,   1, 2, 5, 2, 1, # Female dominated
                3, 3, 3,   7, 2,   2, 1, 3, 1, 2, # Male dominated
                7, 2, 1,   6, 4,   7, 0.001, 2, 0.001, 1  # Many cyclists, young
                )
all.msim <- matrix(all.msim, nrow = 5, byrow = T) # Convert long data into matrix, by row
all.msim <- data.frame(all.msim) # Convert this into a dataframe
names(all.msim) <- category.labels # Add correct column names
all.msim # Show the data frame in R
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000
```

```
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000
```

```
# Check totals for each constraint match
rowSums(all.msim[,1:3]) # Age constraint
```

```
## [1] 10 10 11  9 10
```

```
rowSums(all.msim[,4:5]) # Sex constraint
```

```
## [1] 10 10 11  9 10
```

```
rowSums(all.msim[,6:10])# Mode constraint
```

```
## [1] 10 10 11  9 10
```

Reweighting the survey dataset

Iterative proportional fitting will determine the weight allocated to each individual for each zone to best match the geographically aggregated data. A weight matrix is therefore created, with rows corresponding to individuals and columns to zones.

Create weights: one set of weights for each constraint and one for starting

```
weights0 <- array(dim = c(nrow(USd), nrow(all.msim)))
weights1 <- array(dim = c(nrow(USd), nrow(all.msim)))
weights2 <- array(dim = c(nrow(USd), nrow(all.msim)))
weights3 <- array(dim = c(nrow(USd), nrow(all.msim)))

weights0[, ] <- 1  # sets initial weights to 1
```

Create survey aggregate arrays (for direct comparison with the geographical aggregate data)

```
USd.agg <- array(dim = c(nrow(all.msim), ncol(all.msim)))
USd.agg1 <- array(dim = c(nrow(all.msim), ncol(all.msim)))
USd.agg2 <- array(dim = c(nrow(all.msim), ncol(all.msim)))
USd.agg3 <- array(dim = c(nrow(all.msim), ncol(all.msim)))
colnames(USd.agg1) <- category.labels
```

Convert survey data into wide form

This step allows the individual-level data to be compared with the aggregated data directly

```
USd.cat <- array(rep(0), dim = c(nrow(USd), length(category.labels != 0)))

USd.cat[which(USd$age <= 30), 1] <- 1
USd.cat[which(USd$age >= 31 & USd$age <= 50), 2] <- 1
USd.cat[which(USd$age > 50), 3] <- 1
USd.cat[which(USd$sex == "m"), 4] <- 1
USd.cat[which(USd$sex == "f"), 5] <- 1
USd.cat[which(USd$mode == "bicycle"), 6] <- 1  # Mode constraints
USd.cat[which(USd$mode == "bus"), 7] <- 1
USd.cat[which(USd$mode == "car.d"), 8] <- 1
USd.cat[which(USd$mode == "car.p"), 9] <- 1
USd.cat[which(USd$mode == "walk"), 10] <- 1
sum(USd.cat)  # Should be 30: 3 variables by 10 individuals
```

```
## [1] 30
```

```
for (i in 1:nrow(all.msim)) {
    # Loop creating aggregate values (to be repeated later)
    USd.agg[i, ] <- colSums(USd.cat * weights0[, i])
}

# Test results
USd.agg
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    3    5    2    4    6    1    1    5    1     2
## [2,]    3    5    2    4    6    1    1    5    1     2
## [3,]    3    5    2    4    6    1    1    5    1     2
## [4,]    3    5    2    4    6    1    1    5    1     2
## [5,]    3    5    2    4    6    1    1    5    1     2
```

```
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000
```

```
plot(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg)), xlab = "Constraints",
    ylab = "Model output")
abline(a = 0, b = 1)
```

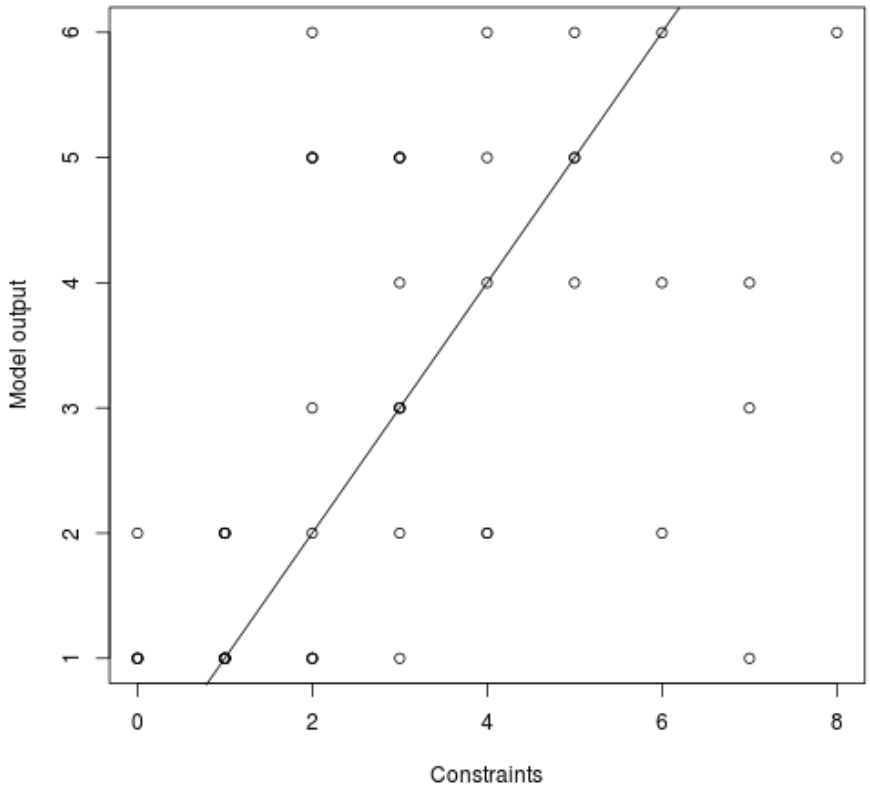


Figure 1: plot of chunk unnamed-chunk-5

```
cor(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg)))
```

```
## [1] 0.546
```

Note that for USd.agg, the results are the same for every zone, as each individual has a weight of 1 for every zone. The next stage is to apply the first constraint, to adjust the weights of each individual so they match the age constraints.

Constraint 1: age

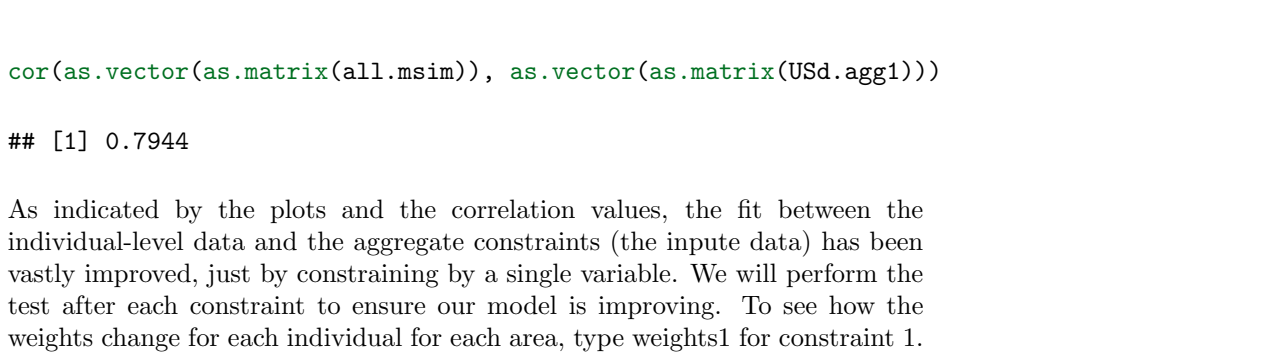
```
for (j in 1:nrow(all.msim)) {
    weights1[which(USd$age <= 30), j] <- all.msim[j, 1]/USd.agg[j, 1]
    weights1[which(USd$age >= 31 & USd$age <= 50), j] <- all.msim[j, 2]/USd.agg[j,
        2]
    weights1[which(USd$age >= 51), j] <- all.msim[j, 3]/USd.agg[j, 3]  ##
}
# Aggregate the results for each zone
for (i in 1:nrow(all.msim)) {
    USd.agg1[i, ] <- colSums(USd.cat * weights0[, i] * weights1[, i])
}
# Test results
USd.agg1
```

```
##      16-30 31-50 50+     m     f bicycle bus car.d car.p walk
## [1,]     3     3   4 4.600 5.400  1.0000 0.6 6.600   0.6  1.2
## [2,]     2     2   6 4.733 5.267  0.6667 0.4 7.733   0.4  0.8
## [3,]     3     4   4 4.800 6.200  1.0000 0.8 6.800   0.8  1.6
## [4,]     3     3   3 4.100 4.900  1.0000 0.6 5.600   0.6  1.2
## [5,]     7     2   1 5.567 4.433  2.3333 0.4 6.067   0.4  0.8
```

```
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000
```

```
plot(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg1)), xlab = "Constraints",
    ylab = "Model output")
abline(a = 0, b = 1)
```



```
cor(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg1)))
```

```
## [1] 0.7944
```

As indicated by the plots and the correlation values, the fit between the individual-level data and the aggregate constraints (the inpute data) has been vastly improved, just by constraining by a single variable. We will perform the test after each constraint to ensure our model is improving. To see how the weights change for each individual for each area, type weights1 for constraint 1.

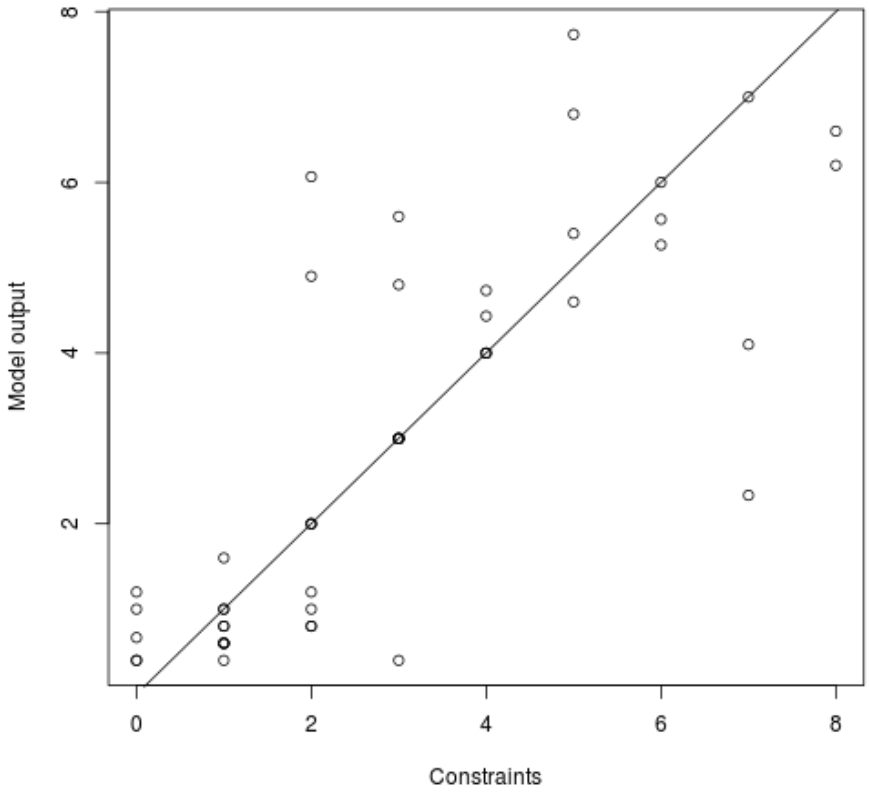


Figure 2: plot of chunk unnamed-chunk-6

```
weights1
```

```
##       [,1]   [,2] [,3] [,4]  [,5]
##  [1,]  1.0 0.6667  1.0  1.0 2.333
##  [2,]  2.0 3.0000  2.0  1.5 0.500
##  [3,]  0.6 0.4000  0.8  0.6 0.400
##  [4,]  0.6 0.4000  0.8  0.6 0.400
##  [5,]  0.6 0.4000  0.8  0.6 0.400
##  [6,]  1.0 0.6667  1.0  1.0 2.333
##  [7,]  2.0 3.0000  2.0  1.5 0.500
##  [8,]  1.0 0.6667  1.0  1.0 2.333
##  [9,]  0.6 0.4000  0.8  0.6 0.400
## [10,]  0.6 0.4000  0.8  0.6 0.400
```

Constraint 2: sex

```
for (j in 1:nrow(all.msim)) {
    weights2[which(USd$sex == "m"), j] <- all.msim[j, 4]/USd.agg1[j, 4]
    weights2[which(USd$sex == "f"), j] <- all.msim[j, 5]/USd.agg1[j, 5]
}

for (i in 1:nrow(all.msim)) {
    USd.agg2[i, ] <- colSums(USd.cat * weights0[, i] * weights1[, i] * weights2[,
        i])
}
# Test results
USd.agg2
```

```
##       [,1]  [,2]  [,3] [,4] [,5]   [,6]   [,7]  [,8]   [,9]  [,10]
## [1,] 3.100 2.874 4.026    5    5 0.9259 0.5556 6.755 0.5556 1.2077
## [2,] 1.886 2.161 5.953    4    6 0.7595 0.4557 7.535 0.4557 0.7937
## [3,] 2.540 4.629 3.831    3    8 1.2903 1.0323 6.113 1.0323 1.5323
## [4,] 3.823 2.004 3.173    7    2 0.4082 0.2449 6.833 0.2449 1.2693
## [5,] 7.135 1.875 0.990    6    4 2.1053 0.3609 6.381 0.3609 0.7920
```

```
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000
```

```r
plot(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg2)), xlab = "Constraints",
    ylab = "Model output")
abline(a = 0, b = 1)
```

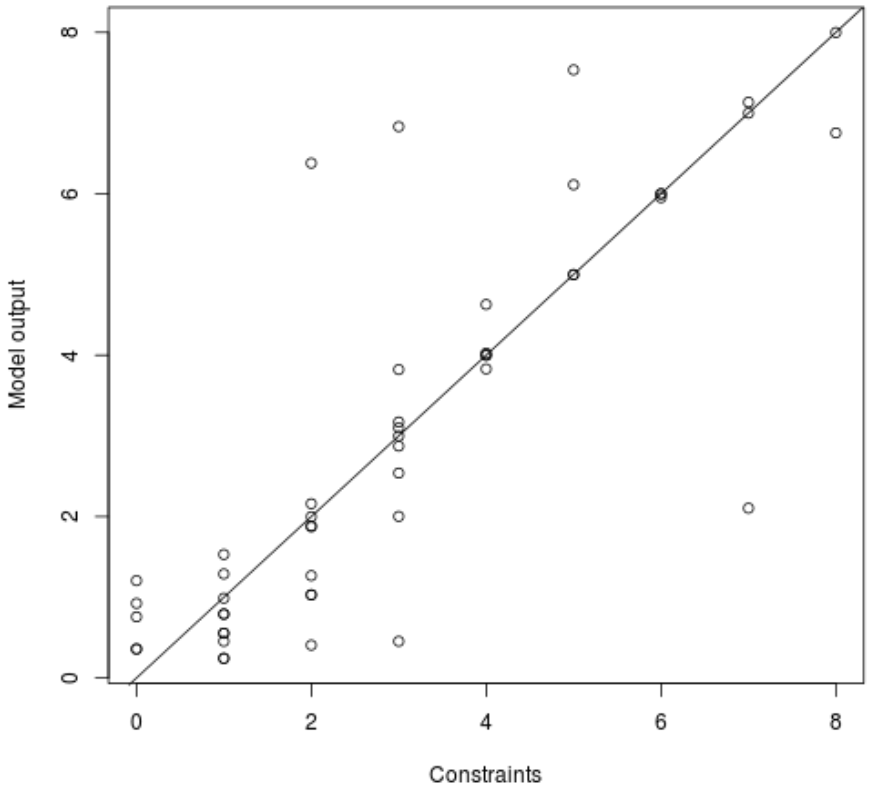


Figure 3: plot of chunk unnamed-chunk-8

```r
cor(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg2)))
```

```
## [1] 0.8362
```

Again the correlation has improved. Now onto the 3rd constraint

Constraint 3: mode

```
for (j in 1:nrow(all.msim)) {
    weights3[which(USd$mode == "bicycle"), j] <- all.msim[j, 6]/USd.agg2[j,
        6]
    weights3[which(USd$mode == "bus"), j] <- all.msim[j, 7]/USd.agg2[j, 7]
    weights3[which(USd$mode == "car.d"), j] <- all.msim[j, 8]/USd.agg2[j, 8]
    weights3[which(USd$mode == "car.p"), j] <- all.msim[j, 9]/USd.agg2[j, 9]
    weights3[which(USd$mode == "walk"), j] <- all.msim[j, 10]/USd.agg2[j, 10]
}
weights4 <- weights0 * weights1 * weights2 * weights3
for (i in 1:nrow(all.msim)) {
    USd.agg3[i, ] <- colSums(USd.cat * weights4[, i])
}
# Test results
USd.agg3
```

```
##        [,1]  [,2]   [,3]  [,4]  [,5]  [,6]  [,7] [,8]  [,9] [,10]
## [1,] 2.5755 2.659 4.7676 5.150 4.852 0.001 1.000    8 1.000 0.001
## [2,] 0.7486 5.302 3.9500 2.856 7.145 0.001 3.000    5 1.000 1.000
## [3,] 2.0224 5.844 3.1332 2.371 8.629 1.000 2.000    5 2.000 1.000
## [4,] 3.4992 4.108 1.3932 4.238 4.762 2.000 1.000    3 1.000 2.000
## [5,] 8.5766 1.115 0.3103 2.290 7.712 7.000 0.001    2 0.001 1.000
```

```
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000
```

```
plot(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg3)), xlab = "Constraints",
    ylab = "Model output")
abline(a = 0, b = 1)
```



```
cor(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg3)))
```

```
## [1] 0.8588
```

```
rowSums(USd.agg3[, 1:3])  # The total population modelled for each zone, constraint 1
```

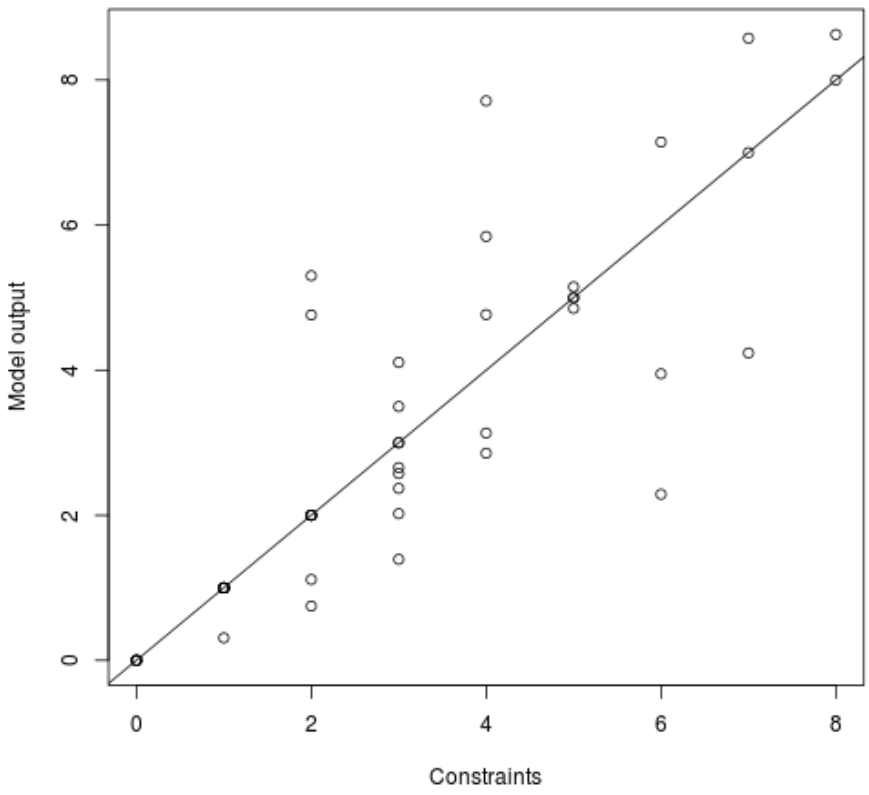


Figure 4: plot of chunk unnamed-chunk-9

```
## [1] 10 10 11  9 10
```

```
rowSums(USd.agg3[, 4:5])
```

```
## [1] 10 10 11  9 10
```

```
rowSums(USd.agg3[, 6:10])
```

```
## [1] 10 10 11  9 10
```

Iterations

The correlation has been improved from one constraint to the next. Even the after the final constraint (mode), which differs greatly from the survey data for some zones, the correlation has improved. This illustrates the robustness of the IPF method. Also note that the population of each simulated zone is correct. The next step is to perform further iterations, using the results of the first iteration (weights4) as our starting point.

```
weights0 <- weights4
USd.agg.1 <- USd.agg  # Saving this for future reference
```

After running this command, simply run the model again, beginning from the loop at the end of the R code section titled "Convert survey data into wide form". After running constraint 3 the second time, the correlation is higher: 0.89 instead of 0.86 . Because this is a relatively simple example, the fit between constraint and simulated aggregate variables will not improve much beyond this point (notice the similarity between the plot below - of the final result after the 2nd iteration - and the plot above).

After the second iteration, the results are as follows:

```
plot(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg3)), xlab = "Constraints",
    ylab = "Model output")
abline(a = 0, b = 1)
```

```
cor(as.vector(as.matrix(all.msim)), as.vector(as.matrix(USd.agg3)))
```

```
## [1] 0.8847
```

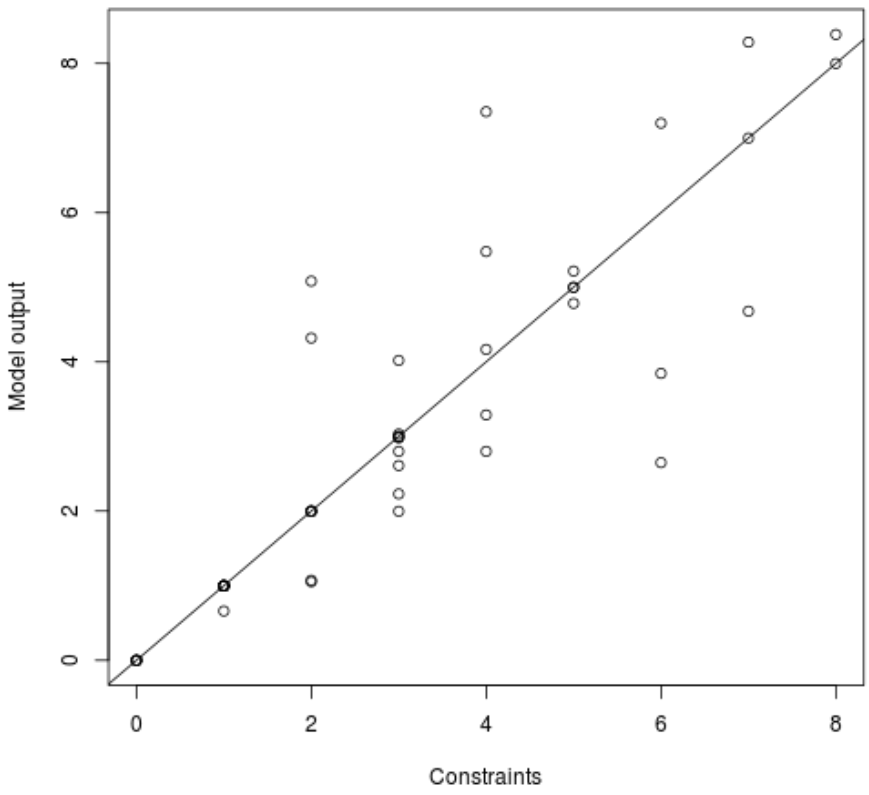


Figure 5: plot of chunk unnamed-chunk-12

Interrogating the results

To view the characteristics of representative individuals for each zone, the vector associated with the zone in question can be called from the final weight matrix (weights4 in this case). The individuals that best match (have the highest weights) for zone five for example, which contains a high proportion of young cyclists, can be viewed with the following command:

```
cbind(weights4[, 5], USd)
```

```
##    weights4[, 5] id age sex    mode
## 1        0.64259  1  27   m   car.d
## 2        0.54367  2  54   m   car.d
## 3        0.00100  3  35   f     bus
## 4        0.82114  4  42   m    walk
## 5        0.00100  5  50   f   car.p
## 6        0.64259  6  19   m   car.d
## 7        0.11842  7  62   f   car.d
## 8        7.00000  8  21   f bicycle
## 9        0.17886  9  38   f    walk
## 10       0.05273 10  48   f   car.d
```

The output from this command illustrates why the final solution of the IPF procedure is not perfect (i.e. why the correlation between the constraint and simulated aggregates cannot be 1): there is only 1 cyclist in the sample population, so she must be replicated 7 times to fulfill the number of cyclists in the constraint variables, distorting the other results. This example demonstrates the importance of having a large and diverse survey dataset from which individuals can be sampled.

For some areas the results are better than others. A breakdown of model fit by area can be seen by recursively running the correlation command:

```
for (i in 1:nrow(all.msim)) {
    all.msim$cor[i] <- cor(as.vector(as.matrix(all.msim[i, 1:10])), USd.agg3[i,
        ])
}
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk    cor
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001 0.9987
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000 0.8016
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000 0.9648
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000 0.7160
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000 0.8580
```

Note that zones 1 and 3 are simulated well. This can be explained because their attributes already fitted with those of original dataset well:

```
for (i in 1:nrow(all.msim)) {
    all.msim$cor[i] <- cor(as.vector(as.matrix(all.msim[i, 1:10])), USd.agg.1[i,
        ])
}
all.msim
```

```
##   16-30 31-50 50+ m f bicycle   bus car.d car.p  walk    cor
## 1     3     3   4 5 5   0.001 1.000     8 1.000 0.001 0.7759
## 2     2     2   6 4 6   0.001 3.000     5 1.000 1.000 0.5728
## 3     3     4   4 3 8   1.000 2.000     5 2.000 1.000 0.8375
## 4     3     3   3 7 2   2.000 1.000     3 1.000 2.000 0.4152
## 5     7     2   1 6 4   7.000 0.001     2 0.001 1.000 0.2112
```

```
USd.agg.1
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    3    5    2    4    6    1    1    5    1     2
## [2,]    3    5    2    4    6    1    1    5    1     2
## [3,]    3    5    2    4    6    1    1    5    1     2
## [4,]    3    5    2    4    6    1    1    5    1     2
## [5,]    3    5    2    4    6    1    1    5    1     2
```

This illustrates the importance of using a sample survey dataset that is fairly representative of the aggregated constraint table.

Taking IPF further

This document has provided a succinct run-through of the IPF procedure in R. This simplified example illustrates how the R programming language is well-suited to the task, with a number of in-built functions to manipulate and analyse the data. Because R is an object-orientated programming language with many add-ons and the ability to define new functions, the capabilities outlined above only scratch the surface of what is possible. For example, it would be possible to create new individuals which have the characteristics most needed to improve the overall model fit. Also, the possibility of grouping individuals into household units would greatly add to the technique's ability to simulate reality (many statistics are collected, and many decisions are made, at the household level). Another 'add-on' that would be particularly useful for IPF would be the ability to 'integerise' the results. Methods for performing integerisation in R are compared in a paper that accompanies this document.

The advancement of the IPF procedure will depend not only on application, but understanding of the underlying theory. This topic is beyond the scope of this practical guide. Some references on the underlying theory and applications of IPF are provided below:

Key references on IPF and spatial microsimultion

Note: this list is just a starter and is no way comprehensive

Deming, W., 1940. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. The Annals of Mathematical Statistics.

Wong, D.W.S., 1992. The Reliability of Using the Iterative Proportional Fitting Procedure. The Professional Geographer, 44(3), pp.340–348.

Norman, P., 1999. Putting Iterative Proportional Fitting (IPF) on the Researcher's Desk. School of Geography, University of Leeds.

Ballas, D. et al., 2005. Geography matters: simulating the local impacts of national social policies, Joseph Roundtree Foundation.

Hermes, K. & Poulsen, M., 2012. A review of current methods to generate synthetic spatial microdata using reweighting and future directions. Computers, Environment and Urban Systems, 36(4), pp.281–290.

Pritchard, D.R. & Miller, E.J., 2012. Advances in population synthesis: fitting many attributes per agent and fitting to household and person margins simultaneously. Transportation, 39(3), pp.685–704.