Iterative proportional fitting for matching on population characteristics between CPRD and DIN-LINK

David A Springate

2015-03-09

Introduction

Iterative Proportional fitting (IPF) is a mathematical scaling method for combining information from two (or more) datasets. It can be used to ensure that a table of data is adjusted so that its marginal (row and column) totals agree with constraining marginal totals obtained from an alternative source. IPF acts as a weighting system whereby the original table values are gradually adjusted through repeated (iterative) calculations to fit these marginal constraints. The resultant table of data is a joint probability distribution of maximum likelihood estimates obtained when the probabilities are convergent within an acceptable (pre-defined) level of tolerance (Norman 1999; Speed 2005). IPF has been proven to converge to the unique maximum likelihood estimates for table cell values, given the user-provided constraints (Fienberg 1970). An advantage of this method is that it allows matching on all variables by adjusting the marginal probabilities whilst preserving the interaction structure of the original table. The algorithm also converges very quickly, typically in less than five iterations, even with multiple variables and cross-tabulated cells.

In this paper, we used IPF to match distributions of patients in CPRD to their corresponding distributions in DIN-LINK according to three matching variables:

1. Cancer site prevalence (9 levels)
2. Year of diagnosis (10 levels)
3. Area deprivation (5 levels)

This is because we observed quite different proportions of patients in the two databases for these variables (See tables 1 and 2 in the main paper) and we wanted to investigate whether the results we obtained were influenced by these different proportions. IPF allowed us to analyse repeated CPRD sub-cohorts matching the marginal proportions of the above three variables with those in the DIN cohort.

The method

We used a two-stage approach where marginal probabilites were first generated by IPF and then applied to a weighted bootstrapping algorithm to generate samples with replacement from the CPRD dataset closely matching the DIN marginal distributions of the above variables.

Stage 1: The IPF algorithm

1. A tolerance, *tol* is set
2. A cross-tabluation of the levels of the three matching variables is constructed (9 x 10 x 5 = 450 cells)
3. A weight is assigned to each of the 450 cells by multiplying together the marginal probabilities from the CPRD cohort of the three matching variables in each cell
4. The marginal CPRD probabilities are recalculated by cross-tabulating the weights
5. New weights are calculated by multiplying the existing weights by the original marginal probabilities in DIN divided by the recalculated CPRD marginal probabilities, for each of the matching variables

6. The maximum absolute difference, D, between the original and new weights is found
7. The original weights are replaced by the new weights
8. If $D > tol$, repeat steps 4-6

Stage 2: The weighted bootstrap

Each patient in the original cohort is assigned a weight, generated from the IPF algorithm, according to their particular combination of matching variables. Then, for each of 10,000 iterations:

1. A sample with replacement of the full original cohort is taken, weighted by the assigned IPF weights generated above
2. Survival analysis is run on the sample to generate hazard ratios in the same way as for the main cohort

After 10,000 replicates, the results are pooled and the median and 2.5 and 97.5 percentiles of the hazard ratios and of the proportions of unmatched variables such as smoking and gender are taken for comparison with the original analysis.

Example

Here follows an illustrative example of the IPF algorithm in `R` code. We want to match the marginal proportions of two three-level variables, A and B from dataset $d1$ to dataset $d2$:

```r
tol = 0.001                              # tolerance for weight difference
d1_A <- c(a1 = 38, a2 = 52, a3 = 55)     # Marginal counts for A in d1
d1_B <- c(b1 = 55, b2 = 42, b3 = 48)     # Marginal counts for B in d1
d2_A <- c(a1 = 80, a2 = 134, a3 = 46)    # Marginal counts for A in d2
d2_B <- c(b1 = 60, b2 = 68, b3 = 132)    # Marginal counts for B in d2

## convert to marginal probabilities:
d1_pA <- d1_A / sum(d1_A)
d1_pB <- d1_B / sum(d1_B)
d2_pA <- d2_A / sum(d2_A)
d2_pB <- d2_B / sum(d2_B)

## Cross-tabulate and assign weights
prob_table <- expand.grid(A = names(d1_pA),
                          B = names(d1_pB))
prob_table$weights <- sapply(1:nrow(prob_table),
                             function(x){
                               d1_pA[[prob_table$A[x]]] *  d1_pB[[prob_table$B[x]]]
                             })

## Add marginal probabilities for target d2 to prob_table:
prob_table$d2_pA <- sapply(prob_table$A, function(x) d2_pA[x])
prob_table$d2_pB <- sapply(prob_table$B, function(x) d2_pB[x])
```

```
## display the table so far:
prob_table
```

```
##    A  B    weights     d2_pA     d2_pB
## 1 a1 b1 0.09940547 0.3076923 0.2307692
## 2 a2 b1 0.13602854 0.5153846 0.2307692
## 3 a3 b1 0.14387634 0.1769231 0.2307692
## 4 a1 b2 0.07590963 0.3076923 0.2615385
## 5 a2 b2 0.10387634 0.5153846 0.2615385
## 6 a3 b2 0.10986920 0.1769231 0.2615385
## 7 a1 b3 0.08675386 0.3076923 0.5076923
## 8 a2 b3 0.11871581 0.5153846 0.5076923
## 9 a3 b3 0.12556480 0.1769231 0.5076923
```

```
## define a weighting function for IPF.
## Note that d1_pA and d1_pB are assigned inside the loop below
reweight <- function(prob_table){
  weights <- prob_table$weights
  for(v in c("A", "B")){
    weights <- weights * (prob_table[[paste0("d2_p", v)]] /
                            prob_table[[paste0("d1_p", v)]])
  }
  weights
}

## Iterate until the weights do not change within the tolerance
iter <- 1 # counter
repeat {
  ## assign marginal probabilities for d1
  prob_table$d1_pA <- sapply(prob_table$A,
                             function(x) xtabs(weights ~ A, data = prob_table)[x])
  prob_table$d1_pB <- sapply(prob_table$B,
                             function(x) xtabs(weights ~ B, data = prob_table)[x])
  ## Recalculate weights
  weights <- reweight(prob_table)
  ## calculate max difference and compare with tol
  diffs <- abs(prob_table$weights - weights)
  max_diff <- which(diffs == max(diffs))
  diff_value <- diffs[max_diff] / prob_table$weights[max_diff]
  if (diff_value > tol){ # if tolerance is not reached, continue
    message("Iteration #", iter, ". difference = ", diff_value)
    iter <- iter + 1
    prob_table$weights <- weights
  } else {
    message("Iteration #", iter, ". difference = ", diff_value, ". Stopping")
    prob_table$weights <- weights
    break
  }
}
```

```
## Iteration #1. difference = 1.20406022416932
## Iteration #2. difference = 2.33367775076681e-16. Stopping
```

```
## The marginal probabilities for d1 and d2 are now within the tolerance bounds:
prob_table
```

```
##    A  B    weights     d2_pA     d2_pB     d1_pA     d1_pB
## 1 a1 b1 0.07100592 0.3076923 0.2307692 0.3076923 0.2307692
## 2 a2 b1 0.11893491 0.5153846 0.2307692 0.5153846 0.2307692
## 3 a3 b1 0.04082840 0.1769231 0.2307692 0.1769231 0.2307692
## 4 a1 b2 0.08047337 0.3076923 0.2615385 0.3076923 0.2615385
## 5 a2 b2 0.13479290 0.5153846 0.2615385 0.5153846 0.2615385
## 6 a3 b2 0.04627219 0.1769231 0.2615385 0.1769231 0.2615385
## 7 a1 b3 0.15621302 0.3076923 0.5076923 0.3076923 0.5076923
## 8 a2 b3 0.26165680 0.5153846 0.5076923 0.5153846 0.5076923
## 9 a3 b3 0.08982249 0.1769231 0.5076923 0.1769231 0.5076923
```

```
## The weights can then be used to adjust the probabilities in a weighted bootstrapping
## algorithm, by assigning the weight matching the combination of A and B for each individual.
```

References

Fienberg, Stephen E. 1970. "An Iterative Procedure for Estimation in Contingency Tables." *The Annals of Mathematical Statistics* 41 (3). Institute of Mathematical Statistics: pp. 907–17. http://www.jstor.org/stable/2239244.

Norman, P. 1999. "Putting Iterative Proportional Fitting on the Researcher's Desk." http://eprints.whiterose.ac.uk/5029/.

Speed, Terry P. 2005. "Iterative Proportional Fitting." In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd. doi:10.1002/0470011815.b2a10027. http://dx.doi.org/10.1002/0470011815.b2a10027.