

Module 5: Fellegi-Sunter Method Applied to RLdata500

Rebecca C. Steorts

Load packages

Let's load packages.

```
library(RecordLinkage)
```

```
## Warning: package 'DBI' was built under R version 4.3.3
```

RLdata500

Let's consider the RLdata500 data set.

```
data(RLdata500)  
#head(RLdata500)  
#head(identity.RLdata500)
```

Comparison Vectors

Why do we build record pairs of comparison vectors?

Answer: Reduce the total number of record comparisons.

Comparison Vectors

How do we build record pairs of comparison vectors?

Answer: Use the `compare.dedup` function.

Comparison Vectors

```
# create comparison vectors  
rpairs <- compare.dedup(RLdata500,  
                        identity = identity.RLdata500)
```

Comparison Vectors

```
# inspect comparison vectors  
rpairs$pairs[1:5,]
```

##	id1	id2	fname_c1	fname_c2	lname_c1	lname_c2	by	bm	bd	is_match
## 1	1	2	0	NA	0	NA	0	1	0	0
## 2	1	3	0	NA	0	NA	0	0	0	0
## 3	1	4	0	NA	0	NA	0	0	0	0
## 4	1	5	0	NA	0	NA	0	0	0	0
## 5	1	6	0	NA	0	NA	0	1	0	0

Blocking

Blocking is the reduction of the amount of data pairs through focusing on specified agreement patterns.

Blocking is a common strategy to reduce computation time and memory consumption by only comparing records with equal values for a subset of attributes, called blocking fields.

Blocking

A blocking specification can be supplied to the `compare` function via the argument `blockfld`.

We will consider a blocking pattern where two records must agree in either the **first component of the first name** or **full date of birth**.

Blocking and Comparison Vectors

```
# blocking and comparison vectors  
rpairs <- compare.dedup(RLdata500,  
                        blockfld = list(1,5:7),  
                        identity = identity.RLdata500)
```

Blocking and Comparison Vectors

```
# inspect comparison vectors  
rpairs$pairs[c(1:3, 1203:1204),]
```

##	id1	id2	fname_c1	fname_c2	lname_c1	lname_c2	by	bm	bd	is_match
## 1	1	174	1	NA	0	NA	0	0	0	0
## 2	1	204	1	NA	0	NA	0	0	0	0
## 3	2	7	1	NA	0	NA	0	0	0	0
## 1203	448	497	1	NA	0	NA	0	0	0	0
## 1204	450	477	1	NA	0	NA	0	0	0	0

Observe that these records agree on first name but not date of birth (as designed).

String Comparators

Recall that string comparators measure the similarity between strings, usually with a similarity measure in the range $[0, 1]$, where 0 denotes maximal dissimilarity and 1 equality.

Examples: Edit and Jaro Winkler

Blocking and String Comparators

```
# blocking on birth day and month  
# use jarowinkler string distance  
rpairsfuzzy <- compare.dedup(RLdata500,  
                             blockfld = c(5,6),  
                             strcmp = TRUE,  
                             strcmpfun = jarowinkler)
```

Blocking on **birth day and month** where the **Jaro Winkler** string comparator is used.

Blocking and String Comparators

```
# inspect first five record pairs  
rpairsfuzzy$pairs[1:5,]
```

##	id1	id2	fname_c1	fname_c2	lname_c1	lname_c2	by	bm	bd	is_match
## 1	2	43	1.0000000	NA	0.9666667	NA	1	1	1	NA
## 2	4	392	0.5777778	NA	0.4833333	NA	1	1	1	NA
## 3	6	328	0.4365079	NA	0.4444444	NA	1	1	0	NA
## 4	7	129	0.0000000	NA	0.4416667	NA	1	1	0	NA
## 5	11	130	0.4476190	NA	0.0000000	NA	1	1	0	NA

Probabilistic record linkage

Probabilistic record linkage relies on the assumption of conditional probabilities concerning comparison patterns.

Recall that we defined the u and m probabilities previously as the following:

$$u_{\tilde{\gamma}} = P(\gamma = \tilde{\gamma} \mid \text{the records are a match})$$

$$m_{\tilde{\gamma}} = P(\gamma = \tilde{\gamma} \mid \text{the records are not a match}).$$

Probabilistic record linkage

The probabilities of the random vector $\gamma = (\gamma_1, \dots, \gamma_n)$ having value $\tilde{\gamma} = (\tilde{\gamma}_1, \dots, \tilde{\gamma}_n)$ conditional on the match status Z can more precisely be defined as follows:

$$u_{\tilde{\gamma}} = P(\gamma = \tilde{\gamma} \mid Z = 0)$$

$$m_{\tilde{\gamma}} = P(\gamma = \tilde{\gamma} \mid Z = 1),$$

where $Z = 0$ stands for a non-match and $Z = 1$ stands for a match.

Probabilistic record linkage

In the Fellegi-Sunter model these probabilities are used to compute weights of the form

$$w_{\tilde{\gamma}} = \log \frac{P(\gamma = \tilde{\gamma} \mid Z = 1)}{P(\gamma = \tilde{\gamma} \mid Z = 0)}.$$

These weights are used in order to discern between matches and non-matches, where there are several ways to estimate the probabilities/weights.

EM algorithm

The EM algorithm is used typically to estimate the weights, where the backbone of this algorithm is described by Haber (1984).

Weight calculation based on the EM algorithm and the method by Contiero et al. (2005) are implemented by functions `emWeights` and `epiWeights`.

Calling `summary` on the result shows the distribution of weights in histogram style.

EM algorithm

```
rpairs <- epiWeights(rpairs)
summary(rpairs)
```

```
##
```

```
## Deduplication Data Set
```

```
##
```

```
## 500 records
```

```
## 1221 record pairs
```

```
##
```

```
## 49 matches
```

```
## 1172 non-matches
```

```
## 0 pairs with unknown status
```

```
##
```

```
##
```

```
## Weight distribution:
```

```
##
```

```
## [0.15,0.2] [0.2,0.25] [0.25,0.3] [0.3,0.35] [0.35,0.4] [0.4,0.45] (0
```

```
##          1011          0          89          30          29          8
```

```
## (0.5,0.55] (0.55,0.6] (0.6,0.65] (0.65,0.7] (0.7,0.75] (0.75,0.8]
```

```
##          1          14          19          10          2          1
```

Computing Weight Thresholds

Discernment between matches and non-matches is achieved by means of computing weight thresholds.

The function `epiClassify` allows the user to specify a threshold.

Computing Weight Thresholds

```
result <- epiClassify(rpairs, 0.55)
summary(result)
```

```
##
## Deduplication Data Set
##
## 500 records
## 1221 record pairs
##
## 49 matches
## 1172 non-matches
## 0 pairs with unknown status
##
##
## Weight distribution:
##
## [0.15,0.2] [0.2,0.25] [0.25,0.3] [0.3,0.35] [0.35,0.4] [0.4,0.45] [0.45,0.5]
##      1011      0      89      30      29      8      7
## [0.5,0.55] [0.55,0.6] [0.6,0.65] [0.65,0.7] [0.7,0.75] [0.75,0.8]
##      1      14      19      10      2      1
##
## 46 links detected
## 0 possible links detected
## 1175 non-links detected
##
## alpha error: 0.061224
## beta error: 0.000000
## accuracy: 0.997543
##
##
## Classification table:
##
##           classification
## true status    N    P    L
##      FALSE 1172    0    0
##      TRUE   3     0   46
```

Error Measures

```
# recall and sensitivity are the same below
```

```
getErrorMeasures(result)
```

```
## $alpha  
## [1] 0.06122449  
##  
## $beta  
## [1] 0  
##  
## $accuracy  
## [1] 0.997543  
##  
## $precision  
## [1] 1  
##  
## $sensitivity  
## [1] 0.9387755  
##  
## $specificity  
## [1] 1  
##  
## $ppv  
## [1] 1  
##  
## $npv  
## [1] 0.9974468
```

Other objects

We can look at many objects of `result` which include

- ▶ `data`
- ▶ `pairs`
- ▶ `frequencies`
- ▶ `type`
- ▶ `Wdatas`
- ▶ `prediction`
- ▶ `threshold`