Module X: fastlink

Rebecca C. Steorts

Reading

- Binette and Steorts (2020)
- Others ??

Probabilistic Entity Resolution

While Fellegi and Sunter (1969) have provided a framework for probabilistic entity resolution, there are few implmentations that scale to large size data sets.

fastlink

- Edmorando et al. (2020) developed fastlink a scalable implementation of the FS method.
- In addition, the authors incorporated auxiliary information such as population name frequency and migration rates.
- The authors used parallelization and hashing to merge millions of records in a near real-time on a laptop computer, and provided open-source software of their proposed methodology.

Agreement Patterns

- Two data sets (A and B) with variables in common
- Agreement value in field a for record pair (i, j)

$$\rho_a(i,j) \;=\; \begin{cases} \texttt{agree} \\ \\ \texttt{disagree} \end{cases}$$

Agreement Patterns

	First	Last	Age	Street
Data set $\mathcal{A}$				
1	James	Smith	35	**Devereux St.**
Data set $\mathcal{B}$				
7	James	Smit	43	**Dvereux St.**
	agree	agree	disagree	**agree**

Agreement Patterns

	First	Last	Age	Street
Data set $\mathcal{A}$				
1	James	Smith	35	**Devereux St.**
Data set $\mathcal{B}$				
7	James	Smit	43	**Dvereux St.**
	agree	agree	disagree	**agree**

Agreement pattern $\gamma(i,j) = \{\gamma_1(i,j), \gamma_2(i,j), \ldots, \gamma_K(i,j)\}$

Agreement Patterns

- We observe agreement patterns $\gamma(i,j)$
- We do not observe the matching status

$$C_{i,j} = \begin{cases} \texttt{non-match} \\ \\ \texttt{match} \end{cases}$$

fastlink Model

$$\begin{aligned} C(i,j) &\overset{\text{iid}}{\sim} \text{Bernoulli}(\mu) \\ \rho(i,j) \mid C(i,j) = \text{non-match} &\overset{\text{iid}}{\sim} \mathcal{F}(\pi_{\text{NM}}) \\ \rho(i,j) \mid C(i,j) = \text{match} &\overset{\text{iid}}{\sim} \mathcal{F}(\pi_{\text{M}}) \end{aligned}$$

Where λ, π_{M}, π_{NM} are estimated via the EM algorithm

fastlink

- Available on CRAN
- We investigate it first on RLdata500

fastlink in R

```r
library(fastLink)
library(RecordLinkage)
```

```
## Loading required package: DBI

## Loading required package: RSQLite

## Loading required package: ff

## Loading required package: bit

##
## Attaching package: 'bit'

## The following object is masked from 'package:base':
##
##     xor

## Attaching package ff

## - getOption("fftempdir")=="/var/folders/bv/xhclmwh90zg08
```

RLdata10000

```
# load RLdata10000
records <- read.table("data/RLdata10000.csv", sep=",", head
head(records, 4)
```

```
##   fname_c1 fname_c2   lname_c1 lname_c2   by bm bd rec_
## 1    FRANK     <NA>    MUELLER     <NA> 1967  9 27
## 2   MARTIN     <NA>    SCHWARZ     <NA> 1967  2 17
## 3  HERBERT     <NA> ZIMMERMANN     <NA> 1961 11  6
## 4     HANS     <NA>    SCHMITT     <NA> 1945  8 14
```

RLdata10000

```r
# Number of unique records
length(unique(records$ent_id))
```

```
## [1] 9000
```

#Linkage Fields

```r
# linkage fields
linkageFields <- c("fname_c1", "lname_c1", "by", "bm", "bd'
```

Exact Matching

```r
# perform exact matching
exact.match <- merge(records, records, by = linkageFields)

# number of self-matches
sum(exact.match$rec_id.x == exact.match$rec_id.y)
```

```
## [1] 10000
```

```r
# number of non-self matches
sum(exact.match$rec_id.x != exact.match$rec_id.y)
```

```
## [1] 16
```

Who are they?

```r
head(exact.match[exact.match$rec_id.x
                 != exact.match$rec_id.y,
                 c(linkageFields)], 4)
```

```
##       fname_c1 lname_c1   by bm bd
## 973     BIRGIT ALBRECHT 1947 12  3
## 974     BIRGIT ALBRECHT 1947 12  3
## 1629 CHRISTINA   FRANKE 1997  8 13
## 1630 CHRISTINA   FRANKE 1997  8 13
```

Preparation

```r
# linkage fields
linkageFields <- c("fname_c1", "lname_c1", "by", "bm", "bd'

# string distance fields
stringDistFields <- c("fname_c1", "lname_c1")

# partial distance fields (fields where we allow
# for agree, disagree, and partially agree)
partialMatchFields <- c("fname_c1", "lname_c1")
```

Run fastlink

```
out <- fastLink(dfA = records,
                dfB = records,
                varnames = linkageFields,
                stringdist.match = stringDistFields, # JW t
                partial.match = partialMatchFields,
                cut.a = 0.94, cut.p = 0.84, # JW cutoffs
                dedupe = FALSE # 1-to-1 match
)
```

```
##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
## If you set return.all to FALSE, you will not be able to
## dfA and dfB are identical, assuming deduplication of a s
## Setting return.all to FALSE.
##
## Calculating matches for each variable.
```

fastlink Objects

fastLink has the following objects as output:

```
names(out)
```

```
## [1] "matches"   "EM"         "patterns"  "posterior" "nob
```

Who is matched

The indices of each matched pair can be found in out$matches

```r
head(cbind(out$matches$inds.a, out$matches$inds.b), 6)
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    3    3
## [4,]    4    4
## [5,] 1957    4
## [6,]    5    5
```

Counting Patterns

- Counts and FS weights for each patterns can be found in `out$EM$patterns.w`
- Legend: 2 = Agree; 1 = Partially Agree; 0 = Disagree

```
tail(out$EM$patterns.w[, 1:7])
```

```
##       gamma.1 gamma.2 gamma.3 gamma.4 gamma.5 counts
## [65,]       2       0       2       2       2     48 10.
## [66,]       0       1       2       2       2     20  3.
## [67,]       2       1       2       2       2    242 15.
## [68,]       0       2       2       2       2     56  8.
## [69,]       1       2       2       2       2     20 12.
## [70,]       2       2       2       2       2  10924 20.
```

Matching Threshold

By default it is 0.85, but it can be easily changed:

```r
out <- fastLink(dfA = records, dfB = records, varnames = li
stringdist.match = stringDistFields, partial.match = partia
cut.a = 0.94, cut.p = 0.84,
threshold.match = 0.90, # Matching threshold dedupe = FALSE
)
```

```
##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
## If you set return.all to FALSE, you will not be able to
## dfA and dfB are identical, assuming deduplication of a s
## Setting return.all to FALSE.
##
## Calculating matches for each variable.
## Getting counts for parameter estimation.
```

Exercises/TODO

- How would you calculate the precision and recall?
- Apply this to a much large data set with blocking. (Make this another module).
- Combine with fastlink and dblink