Module X: fastlink

Rebecca C. Steorts

Reading

- Binette and Steorts (2020)
- Edmorando et al. (2020)
- Fellegi and Sunter (1969)

Probabilistic Entity Resolution

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

Agenda

- We review fastlink, Edmorando et al. (2020)
- We illustrate a toy example on `RLdata10000`

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

- Assume two data sets (A and B) with overlapping variables in common (such as name, gender, address, etc.)
- Define an 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** the 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` package

- Available on CRAN
- We investigate it first on RLdata10000
- We assume no blocking

fastlink in R

```
# load libraries
library(fastLink)
library(RecordLinkage)
```

RLdata10000

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

```
##   fname_c1 fname_c2   lname_c1 lname_c2   by bm bd rec_id ent_id
## 1    FRANK     <NA>    MUELLER     <NA> 1967  9 27      1   3606
## 2   MARTIN     <NA>    SCHWARZ     <NA> 1967  2 17      2   2560
## 3  HERBERT     <NA> ZIMMERMANN     <NA> 1961 11  6      3   3892
## 4     HANS     <NA>    SCHMITT     <NA> 1945  8 14      4    329
```

RLdata10000

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

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

Linkage Fields

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

Exact Matching

```
# 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
```

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

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

Who are the non-self matches?

```
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

```
# 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,
                # JW by default
                stringdist.match = stringDistFields,
                partial.match = partialMatchFields,
                # JW cutoffs
                cut.a = 0.94, cut.p = 0.84,
                # 1-to-1 match
                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
```

fastLink Objects

fastLink has the following objects as output:

```
names(out)
```

```
## [1] "matches"   "EM"        "patterns"  "posterior" "nobs.a"
```

Who is matched?

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

```
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             weig
## [65,]       2       0       2       2       2     48 10.493417296327
## [66,]       0       1       2       2       2     20  4.281720787087
## [67,]       2       1       2       2       2    242 14.922337282587
## [68,]       0       2       2       2       2     56  7.560387428981
## [69,]       1       2       2       2       2     20 12.604947813723
## [70,]       2       2       2       2       2  10924 18.201003924480
```

Matching Threshold

By default the **matching threshold 0.85**, but it can be easily changed:

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

```
##
## ====================
## 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.
```

Loading Packages

```
source("evaluationMetrics.R")
#library(exchangeableER)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:stringdist
##
##     extract
```

```
## The following object is masked from 'package:ff':
##
##     add
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:bit':
```

Setup

```
trueMembership <- records$ent_id
recordIds <- records$rec_id
numRecords <- dim(records)[1]
matches <-
  data.table(cbind(out$matches$inds.a,
                   out$matches$inds.b))
head(matches)
```

```
##       V1 V2
## 1:     1  1
## 2:     2  2
## 3:     3  3
## 4:     4  4
## 5: 1957  4
## 6:     5  5
```

```
dim(matches)[1]
```

```
## [1] 12072
```

True Positives, False Positives, and False Negatives

```
## True Positives, False Positives, and False Negatives:
TP <- sum(records$ent_id[matches$V1]
          == records$ent_id[matches$V2])
FP <- sum(records$ent_id[matches$V1]
          != records$ent_id[matches$V2])
FN <- dim(matches)[1] - TP
```

FDR and FNR

```
## False Discovery Rate
FDR <- round(FP/(FP + TP), 2)
FDR
```

```
## [1] 0.01
```

```
## False Negative Rate
FNR <- round(FN/dim(matches)[1], 2)
FNR
```

```
## [1] 0.01
```

Precision and Recall

```r
## Precision
precision <- round(1 - FDR, 2)

## Recall
recall <- round(1 - FNR, 2)

f1 <- round((2.0*TP)/(2.0*TP+FP+FNR), 2)

cbind(precision, recall, f1)
```

```
##      precision recall f1
## [1,]      0.99   0.99  1
```

TODO: The precision and recall is being done a bit weird here.

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