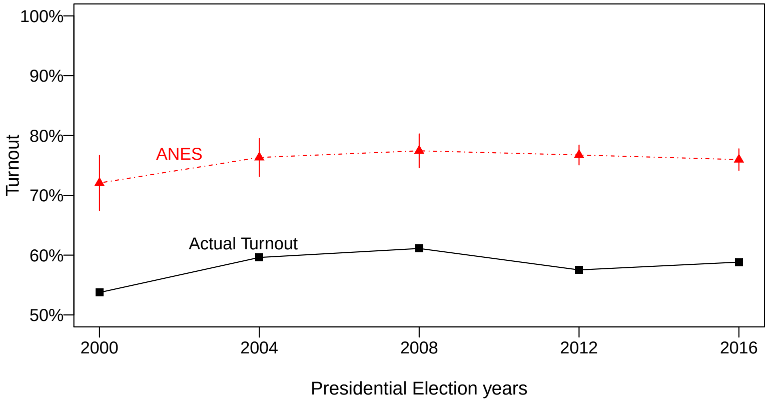
`fastLink`: Probabilistic Record Linkage in R

Ted Enamorado
Washington University in St. Louis

Duke University
February 24, 2021

Social Scientists Rely on Combining Data



- Where does this gap come from?
  - **Over-reporting** vs **Sample of likely-voters**
- At the center of this debate: **Turnout Validation**

  $\rightsquigarrow$ merging datasets without a unique identifier. But How?

How We Merge Data Sets Matters

» **Deterministic methods** e.g., exact matches
  ★ controls false positives
  ⇝ not robust to **typographical errors** and **missing values**

  Georgia's "Exact Match" voter identification law (House Bill 268) ⇝
  In 2018, more than **53,000** records failed to match exactly and were put on hold
  ⇝ **80%** of which are racial minorities

» **Probabilistic Record Linkage**
  ★ robust to typographical errors
  ★ designed to control error rates

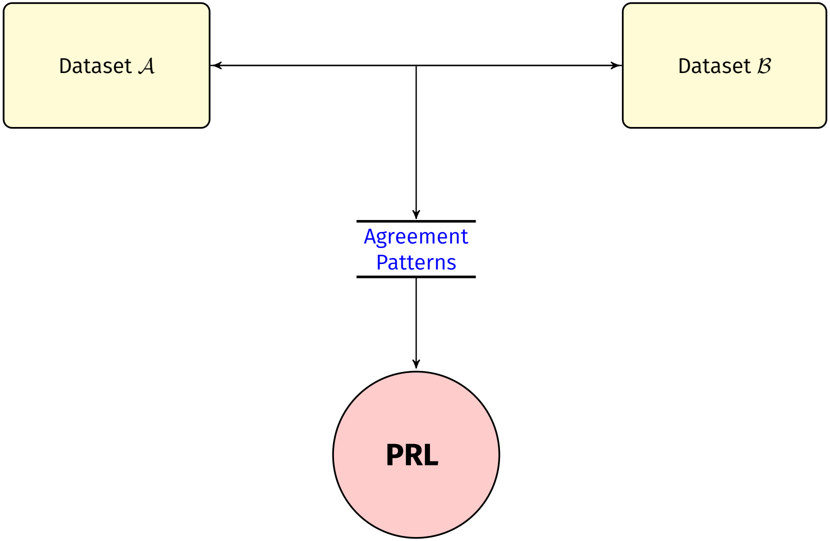
  ⇝ existing open-source implementations of Fellegi-Sunter do not scale
  Solution: `fastLink` (Enamorado, Fifield, and Imai, 2018)

Outline for Today

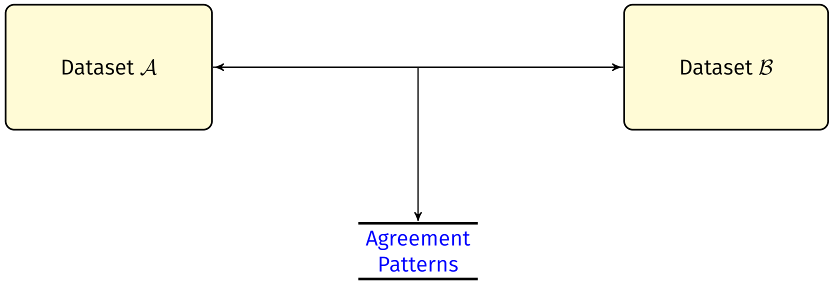
1. **The Method:** The Fellegi-Sunter Model via `fastLink`

2. **Guided example:** Deduplication of `RLdata10000`

The Workflow for PRL à la Fellegi-Sunter



The Workflow for PRL à la Fellegi-Sunter



Agreement Patterns

- Two data sets: $\mathcal{A}$ and $\mathcal{B}$
- K variables in common
- $N_{\mathcal{A}} \times N_{\mathcal{B}}$ pairs to be compared for each common variable k
- Agreement value in field k for a pair (i, j)

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

▸ Agreement value

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

How to Build an Agreement Pattern

	Name			
	First	Last	Age	Street
Data set $\mathcal{A}$				
1	James	Smith	35	Devereux St.
2	John	Martin	56	Devereux St.
Data set $\mathcal{B}$				
1	Michael	Martinez	28	16th St.
2	James	Smitj	34	Dvereuux St.
Agreement patterns $\gamma(i,j)$				
$\gamma(1,1)$	disagree	disagree	disagree	disagree
$\gamma(1,2)$	agree	agree	agree	similar
$\gamma(2,1)$	disagree	similar	disagree	disagree
$\gamma(2,2)$	disagree	disagree	disagree	similar

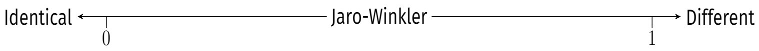
The computational bottleneck: make all these comparisons!

How to Build an Agreement Pattern

Step 1: Calculate a Measure of Dissimilarity

	Name			
	First	Last	Age	Street
$1 \in \mathcal{A}$	`James`	`Smith`	35	`Devereux St.`
$2 \in \mathcal{B}$	`James`	`Smitj`	34	`Dvereuux St.`
`dissimilarity`	`0.00`	`0.04`		`0.10`

» For fields such as first name, last name, street name we use:

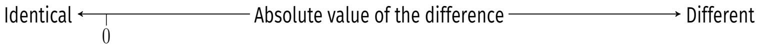


How to Build an Agreement Pattern

Step 1: Calculate a Measure of Dissimilarity

	Name			
	First	Last	Age	Street
$1 \in \mathcal{A}$	James	Smith	35	Devereux St.
$2 \in \mathcal{B}$	James	Smitj	34	Dvereuux St.
dissimilarity			1.00	

» For fields such as age:

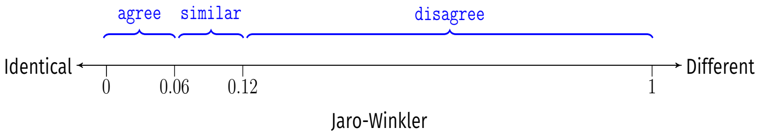


How to Build an Agreement Pattern

Step 2: Make Comparisons Discrete

	Name			
	First	Last	Age	Street
$1 \in \mathcal{A}$	James	Smith	35	Devereux St.
$2 \in \mathcal{B}$	James	Smitj	34	Dvereuux St.
dissimilarity	0.00	0.04		0.10
$\gamma(1,2)$	agree	agree		similar

» Set thresholds for each dissimilarity measure:

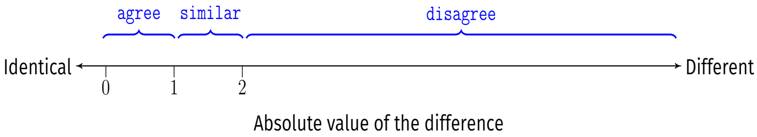


How to Build an Agreement Pattern

Step 2: Make Comparisons Discrete

	Name			
	First	Last	Age	Street
$1 \in \mathcal{A}$	James	Smith	35	Devereux St.
$2 \in \mathcal{B}$	James	Smitj	34	Dvereuux St.
dissimilarity	0.00	0.04	1.00	0.10
$\gamma(1,2)$	agree	agree	agree	similar

» Set thresholds for each dissimilarity measure:

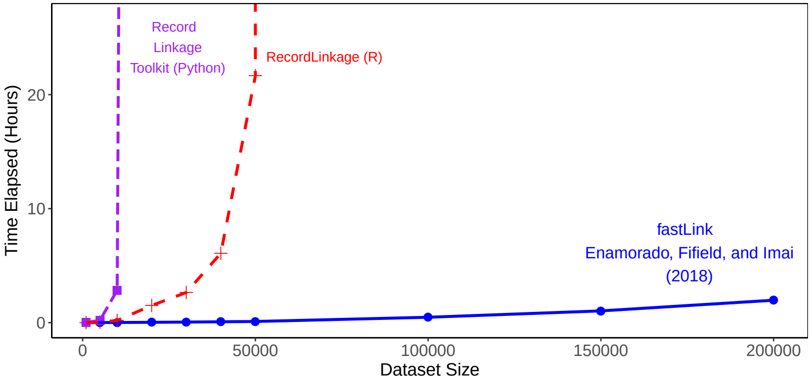


How to Build an Agreement Pattern

	Name			
	First	Last	Age	Street
Data set $\mathcal{A}$				
1	James	Smith	35	Devereux St.
2	John	Martin	56	Devereux St.
Data set $\mathcal{B}$				
1	Michael	Martinez	28	16th St.
2	James	Smitj	34	Dvereuux St.
Agreement patterns $\gamma(i, j)$				
$\gamma(1,1)$	disagree	disagree	disagree	disagree
$\gamma(1,2)$	agree	agree	agree	similar
$\gamma(2,1)$	disagree	similar	disagree	disagree
$\gamma(2,2)$	disagree	disagree	disagree	similar

The computational bottleneck: make all these comparisons!

Runtime Comparison



- Data sets of equal size
- Variables in common:
  first and last name; house number, street name and zip code; age

Computational Improvements:

- Sufficient statistic:

  Count the number of pairs per agreement pattern

- Bottleneck: making the comparisons!

- Solutions:

  1. Agreement values (and consequently agreement patterns) are mutually exclusive
  2. Sparse matrices are used to store agreement values and patterns (hash tables)
  3. Each hash table can be connected to all the others via a new hashing key function
  4. The entire process can be run in parallel

Hashing

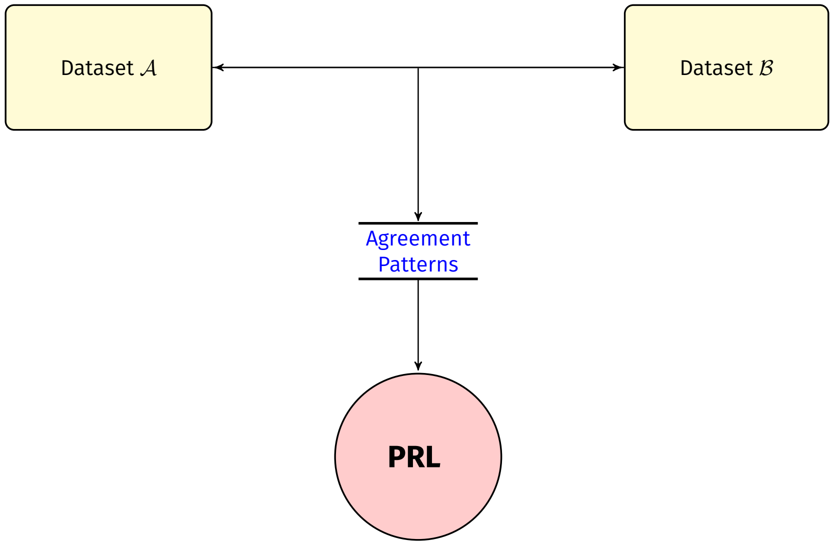
- $\mathbf{H}_k$ maps each pair of records (keys) in linkage field k to a corresponding agreement pattern (hash value):

$$\mathbf{H} = \sum_{k=1}^{K} \mathbf{H}_k \quad \text{where} \quad \mathbf{H}_k = \begin{bmatrix} h_k^{(1,1)} & h_k^{(1,2)} & \dots & h_k^{(1,N_2)} \\ \vdots & \vdots & \ddots & \vdots \\ h_k^{(N_1,1)} & h_k^{(N_1,2)} & \dots & h_k^{(N_1,N_2)} \end{bmatrix}$$

and $h_k^{(i,j)} = \mathbf{1}\{\gamma_k(i,j) > 0\}\, 2^{\gamma_k(i,j) + \mathbf{1}\{k>1\} \times \sum_{e=1}^{k-1}(L_e - 1)}$

- $\mathbf{H}_k$ is a sparse matrix, and so is $\mathbf{H}$
- With sparse matrix, lookup time is $O(T)$ where T is the number of unique patterns observed $T \ll \prod_{k=1}^{K} L_k$

The Workflow for PRL à la Fellegi-Sunter



The Fellegi-Sunter Model of PRL

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

$$M(i,j) = \begin{cases} \text{non-match} \\ \text{match} \end{cases}$$

Mixture Model

$$\begin{aligned} M(i,j) &\overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(\lambda) \\ \gamma(i,j) \mid M(i,j) = \text{non-match} &\overset{\text{i.i.d.}}{\sim} \mathcal{F}(\pi_{\text{NM}}) \\ \gamma(i,j) \mid M(i,j) = \text{match} &\overset{\text{i.i.d.}}{\sim} \mathcal{F}(\pi_{\text{M}}) \end{aligned}$$

The Likelihood of PRL

Independence assumptions:

1. Independence across pairs
2. Independence across linkage fields: $\gamma_k(i,j) \perp\!\!\!\perp \gamma_{k'}(i,j) \mid M(i,j)$

Observed-data log-likehood:

$$\log L(\lambda, \pi \mid \gamma(i,j)) \propto$$

$$\prod_{i=1}^{N_1} \prod_{j=1}^{N_2} \left\{ \lambda \prod_{k=1}^{K} \prod_{l=1}^{L_k - 1} \pi_{Mkl}^{\mathbf{1}\{\gamma_k(i,j)=l\}} + (1-\lambda) \prod_{k=1}^{K} \prod_{l=1}^{L_k - 1} \pi_{NMkl}^{\mathbf{1}\{\gamma_k(i,j)=l\}} \right\}$$

The Likelihood of PRL

From the E-step we obtain:

$$\begin{aligned}
\zeta_{ij} &= \Pr(M(i,j) = \text{match} \mid \lambda, \pi_{kml}, \gamma(i,j)) \\
&= \frac{\lambda \prod_{k=1}^{K} \left(\prod_{l=0}^{L_k - 1} \pi_{k1l}^{\mathbf{1}\{\gamma_k(i,j)=l\}} \right)}{\sum_{m=0}^{1} \lambda^m (1-\lambda)^{1-m} \prod_{k=1}^{K} \left(\prod_{l=0}^{L_k-1} \pi_{kml}^{\mathbf{1}\{\gamma_k(i,j)=l\}} \right)}
\end{aligned}$$

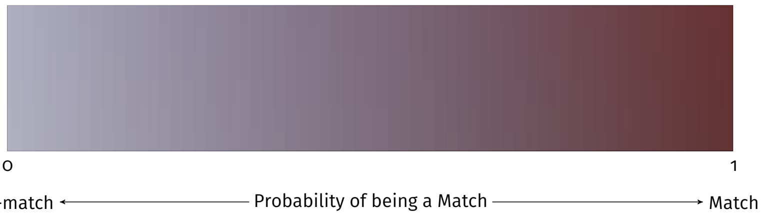
From the M-step:

$$\begin{aligned}
\lambda &= \frac{\sum_{i=1}^{N_{\mathcal{A}}} \sum_{j=1}^{N_{\mathcal{B}}} \zeta_{ij}}{N_{\mathcal{A}} N_{\mathcal{B}}} \\
\pi_{kml} &= \frac{\sum_{i=1}^{N_{\mathcal{A}}} \sum_{j=1}^{N_{\mathcal{B}}} \mathbf{1}\{\gamma_k(i,j) = l)\}(\zeta_{ij})^m (1-\zeta_{ij})^{1-m}}{\sum_{i=1}^{N_{\mathcal{A}}} \sum_{j=1}^{N_{\mathcal{B}}} (\zeta_{ij})^m (1-\zeta_{ij})^{1-m}}
\end{aligned}$$

The Probability of Being a Match

Using parameters of the model + the agreement patterns we can get an estimate of:

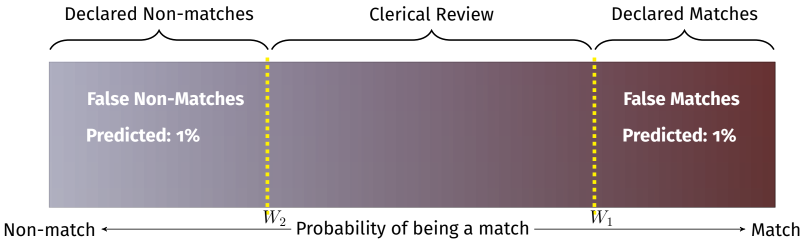
$$\zeta(i,j) = \Pr\left(M(i,j) = \text{match} \,\middle|\, \gamma(i,j), \lambda, \pi_{\text{M}}, \pi_{\text{NM}}\right)$$



The Probability of Being a Match

Using parameters of the model + the agreement patterns we can get an estimate of:

$$\zeta(i,j) = \Pr\left(M(i,j) = \text{match} \,\middle|\, \gamma(i,j), \lambda, \pi_{\mathsf{M}}, \pi_{\mathsf{NM}}\right)$$



Outline

1. **The Method:** The Fellegi-Sunter Model via `fastLink`

2. **Guided example:** Deduplication of `RLdata10000`

Guided Exercise

- We will use the following dataset: `RLdata10000` from the `RecordLinkage` R-package
- Widely used to test PRL methods
- 7 possible linkage fields. We will use 5 of them:
  { first name, last name, birth year, month of birth, day of birth }
- It contains 10,000 records + unique identifier is available for validation
  - 8,000 records have no duplicates
  - 1,000 have a duplicate
- Duplicated records have been perturbed in one field (at random)
- Can we find the duplicates?

Fields:

- `fname_c1`: first name
- `fname_c2`: middle name (if available)
- `lname_c1`: first last name
- `lname_c2`: second last name
- `by`: birth year
- `bm`: birth month
- `bd`: birth day
- `rec_id`: row number
- `ent_id`: unique identifier

RLdata10000

```
## Read data
records <- read.csv("RLdata10000.csv")
```

```
## Let's take a look at the first 4 records in RLdata10000
head(records, 4)

##   fname_c1 fname_c2   lname_c1 lname_c2   by bm bd
## 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
##   rec_id ent_id
## 1      1   3606
## 2      2   2560
## 3      3   3892
## 4      4    329
```

RLdata10000

```r
  ## Count the number of unique ids
  length(unique(records$ent_id))

## [1] 9000
```

```r
## Linkage Fields
linkageFields <- c("fname_c1", "lname_c1",
                   "by", "bm", "bd")

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

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

## [1] 10000
```

RLdata10000

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

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

```r
## Who are they?
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
```

Loading fastLink

```
  ## Load fastLink
  ## Recommended installation: GitHub
  ## https://github.com/kosukeimai/fastLink
  library(fastLink)

## MAC users if you run into GitHub installation
## problems, just send me an email for detailed
## instructions

packageVersion("fastLink")

## [1] '0.6.0'
```

Set Linkage Fields

```r
  ## Linkage Fields (same as above)
  linkageFields <- c("fname_c1", "lname_c1",
                     "by", "bm", "bd")
```

```r
## Fields that we will compare based on
## a string similarity measure
stringDistFields <- c("fname_c1", "lname_c1")
```

```r
## Fields for which we have 3
## possible agreement values
## Agree, Partially Agree, Disagree
partialMatchFields <- c("fname_c1", "lname_c1")
```

Deduplication via `fastLink`

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

```
##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
```

Changing Dissimilarity Measure

```
out <- fastLink(dfA = records, dfB = records,
                varnames = linkageFields,
                stringdist.method = "lv", # Renormalized Levenshtein
                stringdist.match = stringDistFields,
                partial.match = partialMatchFields,
                cut.a = 0.94, cut.p = 0.84,
                dedupe = FALSE)
```

```
##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
```

`fastLink` **Objects**

» The `fastLink` output contains the following objects:

```
names(out)

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

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,]    5    5
## [6,]    6    6
```

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`

```
round(tail(out$EM$patterns.w[, 1:7]), 3)

##        gamma.1 gamma.2 gamma.3 gamma.4 gamma.5 counts
## [63,]        2       0       2       2       2    286
## [64,]        0       1       2       2       2     16
## [65,]        2       1       2       2       2    264
## [66,]        0       2       2       2       2    274
## [67,]        1       2       2       2       2    414
## [68,]        2       2       2       2       2  10016
##        weights
## [63,]   12.573
## [64,]    7.044
## [65,]   18.081
## [66,]    9.337
## [67,]   14.910
## [68,]   20.374
```

Matching Threshold

» By default it is `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,
                threshold.match = 0.90,
                dedupe = FALSE)

##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
```

Deduplication via `fastLink`

```
  ## Deduplicated Dataset:
  ## If a record is duplicated according to
  ## the results from fastLink, then the duplicated
  ## record(s) will have the same id
  ## The new unique id is called dedupe.ids

  recordsfL <- getMatches(dfA = records, dfB = records,
                          fl.out = out)

## Let's count how unique records
## fastLink finds:
length(unique(recordsfL$dedupe.ids))

## [1] 8983
```

Deduplicated data

```
## The deduplicated data looks like this:
head(recordsfL, 4)
```

```
##   fname_c1 fname_c2   lname_c1 lname_c2   by bm bd
## 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
##   rec_id ent_id dedupe.ids
## 1      1   3606          1
## 2      2   2560       1051
## 3      3   3892       2047
## 4      4    329       3033
```

Inspecting Duplicates

```
  ## Some examples of known duplicates
  recordsfL[recordsfL$ent_id == 20, ]

##      fname_c1 fname_c2  lname_c1 lname_c2   by bm bd
## 2461     PAUL     <NA>   SCHAEFER     <NA> 1942  7  2
## 3612     PAUL     <NA> SCHAECFER     <NA> 1942  7  2
##      rec_id ent_id dedupe.ids
## 2461   2461     20       1505
## 3612   3612     20       1505

recordsfL[recordsfL$ent_id == 77, ]

##      fname_c1 fname_c2 lname_c1 lname_c2   by bm bd
## 1056     JENS     <NA>   WAGNER     <NA> 1976  8 14
## 2763     JENS     <NA>   WAGNER     <NA> 1986  8 14
##      rec_id ent_id dedupe.ids
## 1056   1056     77         66
## 2763   2763     77         66
```

Confusion Table

```
  recordsfL$dupfL <- ifelse(duplicated(recordsfL$dedupe.ids),
                            "Duplicated", "Not duplicated")
recordsfL$dupTrue <- ifelse(duplicated(recordsfL$ent_id),
                            "Duplicated", "Not duplicated")

confusion <- table("fastLink" = recordsfL$dupfL,
                   "True" = recordsfL$dupTrue)
confusion

##                 True
## fastLink         Duplicated Not duplicated
##   Duplicated            982             35
##   Not duplicated         18           8965
```

False Discovery and False Negative Rates

```
  # True Positives, False Positives, and False Negatives:
  TP <- confusion[1, 1]
FP <- confusion[1, 2]
FN <- confusion[2, 1]

# False Discovery Rate:
FDR <- round(FP/(FP + TP), 3)
FDR

## [1] 0.034

# False Negative Rate:
FNR <- round(FN/1000, 3)
FNR

## [1] 0.018
```

False Precision and Recall

```
  ## Precision:
  PRE <- 1 - FDR
PRE

## [1] 0.966

## Recall
REC <- 1 - FNR
REC

## [1] 0.982
```

Comparisons for Numeric Variables

```r
  ## We can add numeric comparisons using dissimilarity
  numericMatchFields <- c("by")

## Make sure these are of class numeric
records$by <- as.numeric(records$by)
```

Adding `numeric.match` as an Option

```
out2 <- fastLink(dfA = records, dfB = records,
                 varnames = linkageFields,
                 stringdist.match = stringDistFields,
                 cut.a = 0.94, cut.p = 0.84,
                 numeric.match = numericMatchFields,
                 cut.a.num = 1.5,
                 partial.match = partialMatchFields,
                 threshold.match = 0.90,
                 dedupe = FALSE)
```

```
##
## ====================
## fastLink(): Fast Probabilistic Record Linkage
## ====================
##
```

How Well Did We Do?

```
## Deduplicated data with numeric comparison for birth year
  recordsfL2 <- getMatches(dfA = records, dfB = records,
                           fl.out = out2)

## Let's count how unique records
## fastLink finds:
length(unique(recordsfL2$dedupe.ids))

## [1] 8596
```

Blocking

- In `fastLink` blocking is seen as a preprocessing step
- The latter means that blocking happens outside the wrapper function
- The idea then is to subset, and then use the wrapper function in each block
- But how?

Traditional Blocking

```r
  ## For illustration purposes:
  ## Pool together those cases with values outside
  ## a "normal" range:
  records$by2 <- records$by
records$by2[records$by < 1924] <- 1923
records$by2[records$by > 2008] <- 2009

## Traditional Blocking:
blockby <- blockData(records, records, varnames = "by2")

##
## ====================
## blockData(): Blocking Methods for Record Linkage
## ====================
##

linkageFields2 <- c("fname_c1", "lname_c1", "bm", "bd")
```

Looping Across Blocks

```r
results <- list()

for(j in 1:length(blockby)) {
  records.temp <- records[blockby[[j]]$dfA.inds, ]

  out.temp <- fastLink(dfA = records.temp, dfB = records.temp,
                       varnames = linkageFields2,
                       stringdist.match = stringDistFields,
                       partial.match = partialMatchFields,
                       cut.a = 0.92, cut.p = 0.84,
                       threshold.match = 0.90,
                       dedupe = FALSE)

  records.temp <- getMatches(dfA = records.temp,
                             dfB = records.temp,
                             fl.out = out.temp)

  records.temp$dedupe.ids <-  paste0("B", j, "_",
                                     records.temp$dedupe.ids)

  results[[j]] <- records.temp
}

##
```

How Many Unique Entities?

```
  recordsfL.blockE <- do.call('rbind', results)

length(unique(recordsfL.blockE$dedupe.ids))

## [1] 9145
```

Concluding Remarks

- Merging data sets is at the core of social science research
- `fastLink` has proven to be successful in many empirical projects of large scale
  - Matching voter files and public opinion surveys
  - Deduplicating campaign contribution data
  - Liking arrest data across years
  - etc.
- New functionality is coming soon:
  - New blocking functions and dissimilarity measures
  - Human annotations to improve precision (Active Learning)
- Please use it! `fastLink` users are our greatest asset!
- Comments and suggestions are always welcome!
  `ted.enamorado@gmail.edu`

Appendix

Additional Results:

Sections:

Agreement Value:

Formally:

$$\gamma_k(i,j) = \begin{cases} 0 & \text{if } S_k(i,j) < \tau_0 \\ 1 & \text{if } S_k(i,j) \in (\tau_0, \tau_1] \\ 2 & \text{if } S_k(i,j) \in (\tau_1, \tau_2] \\ \vdots & \\ L_{k-1} & \text{if } S_k(i,j) > \tau_{L_k-1} \end{cases}$$

Where L_k is the number of categories and $S_k(i,j)$ is a similarity measure. ▸ Go back

Jaro-Winkler:

$$D(s_1, s_2) \;=\; \{J(s_1, s_2) + \ell \cdot w \cdot (1 - J(s_1, s_2))\}$$

where

$$J(s_1, s_2) \;=\; \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t/2}{m}\right) & \text{otherwise} \end{cases}$$

where:

- $|s|$ represents that length of string s
- m is the number of characters in common
- t is the number of transpositions between the common characters
- $\ell \in [0, 4]$ # of consecutive characters in common at the beginning
- $w \in [0, 0.25]$ is the weight given to ℓ

▸ Go back