

## Module 8: Bayesian Fellegi and Sunter

Rebecca C. Steorts

# Reading

- ▶ Binette and Steorts (2020)
- ▶ Sadinle (2014)

## Load Libraries

```
#install.packages("exer_0.2.0.tar.gz", repos = NULL, type=  
library(exer)
```

# Duplicate detection

**Duplicate detection** is the task of finding sets of records that refer to the same entities within a data file.

# Overview of Bayesian Fellegi and Sunter

Give an overview of the framework

# Notation

Assume there are a total of  $n$  records in a database.

Assume there is one database with  $r$  records labeled

$$\{1, 2, \dots, r\}$$

where more than one record can refer to the same entity.

Assume that  $n \leq r$ .

Thus, we can view this problem as partitioning the database into  $n$  groups of matches/non-matches.

# Representation of partitions

A partition of a set is a collection of nonempty and non-overlapping subsets whose union is the original set.

Sadinle (2014) refers such subsets **groups** or **cells**.

## Example

Suppose the database has five records total  $\{1, 2, 3, 4, 5\}$ .

One potential partition can be represented by the following three groups:

$$\{1, 3\}, \{2\}, \{4, 5\}.$$

Each group represents an underlying entity.

In this example, records 1,3 are co-referent; records 4,5 are co-referent, and record 2 is a singleton record.



## Co-reference matrix

A partition can also be represented by a matrix.

Consider the matrix  $\Delta$  of dimension  $r \times r$ ,

where

$$\Delta_{ij} = \begin{cases} 1, & \text{if records } i,j \text{ are co-referent} \\ 0, & \text{otherwise.} \end{cases}$$

$\Delta$  is referred to as the co-reference matrix.

$\Delta$  is symmetric with only ones in the diagonal.

## Labellings of the partition's groups

Unfortunately, it is not computationally inefficient to utilize the co-reference matrix in practice.

An alternative is to use arbitrary labelings of the **partition's groups**.

## Labellings of the partition's groups

Assume that  $r$  the maximum number of entities possibly represented in the database.

Define

$$Z_i = q, \quad i = 1, \dots, r$$

if record  $i$  represents entity  $q$ ,  $1 \leq q \leq r$ .

$$Z = (Z_1, Z_2, \dots, Z_r)$$

contains all the records labels.

Thus,

$$\Delta_{ij} = I(Z_i = Z_j).$$

## Example (Continued)

Recall our database has  $\{1, 2, 3, 4, 5\}$  records and the partition can be represented by the three groups:

$$\{1, 3\}, \{2\}, \{4, 5\}.$$

What would be examples  $Z$  of arbitrary labellings for this example?

## Example (Continued)

The labelings

$$Z = (1, 2, 1, 3, 3)$$

or

$$Z = (4, 1, 4, 2, 2)$$

would correspond to this partition because both  $Z_1 = Z_3 = Z_4 = Z_5$  and  $Z_2$  gets its own value.

## Connection to the $r$ th Bell number

The number of ways in which a data file with  $r$  records can be partitioned is given by the  $r$ th Bell number, which grows rapidly with  $r$ .

See Rota (1964).

## Comparison data

- ▶ Comparison data are obtained by comparing pairs of records, with the goal of finding evidence of whether two records refer to the same entity.
- ▶ Intuitively, two records referring to the same entity should be very similar.

# Notation

Assume  $f$  features.

Compare features  $f$  of records  $(i, j)$  by computing some similarity measure  $S_f(i, j)$ .

The range of  $S_f(i, j)$  is divided into  $L_f + 1$  intervals

$$l_{f0}, l_{f1}, \dots, l_{fL_f},$$

which represent levels of disagreement.

By convention,  $l_{f0}$  denotes the **highest level of agreement** and  $l_{fL_f}$  denotes the **lowest level of agreement**.



# Notation

For records  $(i, j)$  define

$$\gamma_{ij}^f = \ell \quad \text{if} \quad S_f(i, j) \in I_{f\ell}.$$

- ▶ The larger the value of  $\gamma_{ij}^f$  the large the disagreement between records  $(i, j)$  with respect to feature  $f$ .
- ▶ These feature comparisons are collected into a vector for each record pair denoted by

$$\gamma_{ij} = (\gamma_{ij}^1, \gamma_{ij}^2, \dots, \gamma_{ij}^F),$$

which denotes the comparison vector for records  $(i, j)$ , where  $F$  is the number of features being compared.

# Model for the Comparison Data

Assume that the comparison vector  $\gamma_{ij}$  is a realization of a random vector  $\mathbf{\Gamma}_{ij}$ .

The set of record pairs is composed of two types – co-referent and non co-referent record pairs.

# Model for the Comparison Data

One expects the distribution of the comparison vectors  $\Gamma_{ij}$  of co-referent/non co-referent record pairs to be quite different.

For example, one expects to observe more agreements among co-referent pairs than among non-coreferent pairs.

Similarly, one expects many more disagreements among non-coreferent pairs than among co-referent pairs.

# Model for the Comparison Data

This intuition can be formalized by assuming that the distribution of  $\Gamma_{ij}$  is the same for all record pairs that refer to the same entity and is the same for all record pairs that refer to different entities.

This is modeled as follows:

$$\Gamma_{ij} \mid \Delta_{ij} = 1 \stackrel{iid}{\sim} G_1 \quad (1)$$

$$\Gamma_{ij} \mid \Delta_{ij} = 0 \stackrel{iid}{\sim} G_0 \quad (2)$$

for all  $i, j \in P$ , where  $G_1, G_0$  represent the models of the comparison vectors for pairs that are coreferent and noncoreferent, respectively.

## Prior distribution on the coreference partition

The co-reference matrix represents a partition of the entries of  $\Delta$ .

Let  $D$  represent the set of possible co-reference partitions.

The author assumes a uniform prior that assigns equal probability to each partition in  $D$ .

This can be written as

$$\pi(\Delta) \propto I(\Delta \in D).$$

## Prior distribution on the coreference partition

One simple way to obtain this prior for  $Z$  is to assign equal probability to each of the  $r!/(r - n)!$  labelings of a partition with  $n$  cells/groups.

This leads to the prior

$$p(\mathbf{Z}) \propto \frac{(r - n(\mathbf{Z}))}{r!} I(\mathbf{Z} \in Z)$$

where  $n(\mathbf{Z})$  measure the number of different labelings of  $\mathbf{Z}$ .

## A model for independent comparison fields

We describe a simple parametrization for  $G_1$  and  $G_0$ .

This model assumes that the comparison fields are independent for both co-referent and non co-referent records.

Assuming that  $\Gamma_{ij}^f$  takes  $L_f + 1$  values corresponding to levels disagreement, one can model

$$\Gamma_{ij}^f \mid \mathbf{m}_f \sim \text{Multinomial}(\mathbf{m}_f),$$

where

$$\mathbf{m}_f = (m_{f0}, m_{f1}, \dots, m_{f,L_f-1}).$$

## A model for independent comparison fields

Similarly,

$$\Gamma_{ij}^f \mid \mathbf{u}_f \sim \text{Multinomial}(\mathbf{u}_f),$$

where

$$\mathbf{u}_f = (u_{f0}, u_{f1}, \dots, u_{f,L_f-1}).$$



## A model for independent comparison fields

Following the notation of Sadinle (2014),

$\Phi_0 = (u_1, \dots, u_F)$  and  $\Phi_1 = (m_1, \dots, m_F)$  such that

$\Phi_f = (m_f, u_f)$ .

## Prior specification for the model parameters

Sadinle (2014) specified that

$$m_{f\ell} \sim \text{TruncatedBeta}(\alpha_{f\ell}^1, \beta_{f\ell}^1, \lambda_{f\ell}, 1)$$

for  $\ell = 2, \dots, L_f - 1$ .

In addition,

$$m_{f0} \sim \text{Beta}(\alpha_{f0}^1, \beta_{f0}^1).$$

# Prior specification for the model parameters

Sadinle (2014) specified that

$$u_{f\ell} \sim \text{Uniform}(0, 1)$$

for all features and disagreement.

The author stated that one might take

$$u_{f\ell} \sim \text{Beta}(\alpha_{f0}^0, \beta_{f0}^0)$$

if prior information is available.

# Gibbs sampling

In order to approximate the joint posterior of  $\mathbf{Z}$  and  $\Phi$ , one must use a Gibbs sampler.

## sadinle14 package

- ▶ Marchant et. al (2020) have provided a package for implementing Sadinle (2014).
- ▶ We investigate it on the RLdata500 data set.

## sadinle14 package

```
library(magrittr)
library(sadinle14)
library(exer)
# >> INSERT
library(comparator)
# <<
library(clevr)
RLdata500[['rec_id']] <- seq.int(length.out=nrow(RLdata500))
head(RLdata500)
```

##	fname_c1	fname_c2	lname_c1	lname_c2	by	bm	bd	rec_id
## 1	CARSTEN	<NA>	MEIER	<NA>	1949	7	22	1
## 2	GERD	<NA>	BAUER	<NA>	1968	7	27	2
## 3	ROBERT	<NA>	HARTMANN	<NA>	1930	4	30	3
## 4	STEFAN	<NA>	WOLFF	<NA>	1957	9	2	4
## 5	RALF	<NA>	KRUEGER	<NA>	1966	1	13	5
## 6	JUERGEN	<NA>	FRANKE	<NA>	1929	7	4	6

## distance functions

```
# scoringFns <- list(  
#   fname_c1 = function(x, y) dist_NormLevenshtein(x, y, return.matrix  
#   lname_c1 = function(x, y) dist_NormLevenshtein(x, y, return.matrix  
#   by = function(x, y) dist_AbsoluteDifference(x, y, return.matrix = F  
#   bm = function(x, y) dist_AbsoluteDifference(x, y, return.matrix = F  
#   bd = function(x, y) dist_AbsoluteDifference(x, y, return.matrix = F  
# )  
# >> INSERT  
scoringFns <- list(  
  fname_c1 = comparator::Levenshtein(normalize=TRUE),  
  lname_c1 = comparator::Levenshtein(normalize=TRUE),  
  by = function(x, y) abs(x - y),  
  bm = function(x, y) abs(x - y),  
  bd = function(x, y) abs(x - y)  
)  
# <<
```

## breaks

For each scoring function above, we provide a breaks vectors which specifies the discrete levels of agreement (from 'high' agreement to 'low').

```
scoringBreaks <- list(  
  fname_c1 = c(-Inf, .05, .15, .3, Inf),  
  lname_c1 = c(-Inf, .05, .15, .3, Inf),  
  by = c(-Inf, 0, 1, 3, Inf),  
  bm = c(-Inf, 0, 1, 3, Inf),  
  bd = c(-Inf, 0, 2, 7, Inf)  
)
```



## comparison vectors

- ▶ Now we are ready to compute the attribute comparison scores for the record pairs.
- ▶ Since this is a small data set, we consider all pairs using the `computePairs_all` function.
- ▶ For larger data sets, blocking/indexing is recommended using `computePairs_Hamming` or `computePairs_AttrDist`.

## comparison vectors

```
pairs <- RLdata500 %>%  
  computePairs_all(id.col = 'rec_id') %>%  
  computeScores(scoringFns) %>%  
  discretizeScores(scoringBreaks)
```

## speeding up inference

- ▶ To speed up inference, we only consider a subset of the pairs as candidate matches.
- ▶ Specifically, we consider pairs that have a strong agreement on name (accounting for missing names).

## speeding up inference

```
pairs[['candidate']] <-  
  (pairs$fname_c1 < 4) &  
  (pairs$lname_c1 < 4) |  
  is.na(pairs$fname_c1) |  
  is.na(pairs$lname_c1)
```

## priors on $m$ and $u$ probabilities

- ▶ Next we specify the priors on the  $m^*$  and  $u^*$  probabilities for each attribute and agreement level.
- ▶ `lambda` contains the lower truncation points for the truncated Beta priors on the  $m^*$  probabilities. `alpha1` and `beta1` are the shape parameters for the truncated Beta distributions.
- ▶ For simplicity, we specify a flat (uniform) prior. Note: the priors on the  $u^*$  probabilities are uniform by default.

## priors on $m$ and $u$ probabilities

```
lambda <- list(  
  fname_c1 = c(0.8,0.85,0.99),  
  lname_c1 = c(0.8,0.85,0.99),  
  by = c(0.8,0.85,0.99),  
  bm = c(0.8,0.85,0.99),  
  bd = c(0.8,0.85,0.99)  
)  
  
alpha1 <- lapply(lambda, function(x) rep(1, length(x)))  
beta1 <- alpha1
```

## intialization and Gibbs sampler

Finally we initialize the model and run inference using Markov chain Monte Carlo.

```
model <- sadinle14::initializeModel(pairs,
  lambda, alpha1,
  beta1, id.cols
  = c("rec_id.x", "rec_id.y"),
  candidate.col = "candidate")

result <- sadinle14::runInference(model,
  100, thinningInterval=10,
  burninInterval=100)
```

## posterior samples of linkage structure

The posterior samples of the linkage structure (deduplication) are accessible from `result@history$linkage`.

However the linkage structure only includes the record in pairs.

We can obtain samples of the complete linkage structure (for all record) using the following function.

```
linkageChain <- sadinle14::completeLinkageChain(result,  
  RLdata500$rec_id)
```



## evaluations

- ▶ We evaluate the quality of the posterior linkage structure (deduplication) based on a point estimate computed using the *shared most probable maximal matching sets method* from Steorts, Hall, Fienberg (2016).
- ▶ This functionality is available from the `exer` package.

## evaluations

```
predClusters <- exer::smp_clusters(linkageChain)
predMatches <- clevr::clusters_to_pairs(predClusters)
trueMatches <- clevr::membership_to_pairs(identity.RLdata500)
numRecords <- nrow(RLdata500)
```

## evaluations

```
clevr::eval_report_pairs(trueMatches, predMatches,  
                          num_pairs = numRecords*(numRecords-1)/2)
```

```
## $precision  
## [1] 0.9615385  
##  
## $recall  
## [1] 1  
##  
## $specificity  
## [1] 0.999984  
##  
## $sensitivity  
## [1] 1  
##  
## $f1score  
## [1] 0.9803922  
##  
## $accuracy  
## [1] 0.999984  
##  
## $balanced_accuracy  
## [1] 0.999992
```