RESEARCH REPORT SERIES
(*Computing #2002-01)*

BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage

William E. Yancey

Statistical Research Division
U.S. Bureau of the Census
Washington D.C. 20233

Report Issued: March 6, 2002

Abstract

We present documentation for running the **BigMatch** program, a new record linkage tool for use in matching a very large file against a moderate size file. For each of several blocking criteria, the program can extract likely matching record candidates from the large file without requiring sorting of either file.

Key words: record linkage, software

Background

The **BigMatch** program is based on the Census Bureau record linkage program by Winkler *et al*. For this and other record linkage programs, record pairs from two files are brought together to be compared when they agree on a specified blocking criterion. In order for the best matching pairs to be selected, the files must first be sorted according to this blocking criterion. In order for the most matches to be found, record linkage projects generally require successive passes with the application of several blocking criteria, each of which requiring its own file sorts. When one of the files is very large, the time required to sort the file can become prohibitive.

The **BigMatch** program allows one to run several different blocking criteria for a very large file and a moderate size file requiring the large file to be read just once and without requiring sorting either file. Required sorting is done on a key list in memory and the time required to run the program is generally modest. For each blocking criterion, the program outputs a file of records from the large file that are plausible matches to records in the moderate file.

The standard Census Bureau record linkage program features one-to-one matching that results in each record being paired with its most likely match within its blocking group. The **BigMatch** program does not do this, so that an output file may contain several records from the large file that were scored as likely matches to the same record in the moderate file. The purpose of the **BigMatch** program is to function as a preprocessor that can efficiently extract smaller files from a very large file so that these smaller files can be used efficiently with standard record linkage software.

Further information about the record linkage software can be obtained from `william.e.yancey@census.gov` or `william.e.winkler@census.gov`.

Extracting Probable Matches from a Large File for Record Linkage

Program Overview

The purpose of the **bigmatch** program is to extract subfiles of plausible match records from a very large file by making only one sequential pass through the very large file. The large file never has to be sorted. Suppose that we want to find matching records in a very large master file, which we will designate as the *A* file, that correspond to records in a moderately sized file, which we will designate as the *B* file. We define a file to be of moderate size if it can comfortably fit into core memory. The **bigmatch** program allows the user to specify several blocking criteria and the matching field parameters. The program reads in the *B* file and creates a table of the keys for each blocking criterion. The program then proceeds to read in records from the large *A* file. For each blocking criterion, the program determines whether there are any records from the *B* file which have keys that match the key of the *A* file record. For all *B* records with matching key value, a matching comparison weight is computed with the *A* record. If any of the comparison weights exceeds a cutoff value, the *A* record is written out to a subfile corresponding to that blocking criterion.

Running the Program

Compiling the Program

The program `bigmatch.c` can be compiled using a C compiler. On a UNIX machine, one can use the command

```
cc –o bigmatch bigmatch.c –lm
```

where the –lm flag instructs the compiler to link with the C math function library. One may also use optimization flags to enhance performance. The program **bigmatch** currently compiles and runs on Compaq Alphas, Sun workstations, and Windows PCs.

Program Input

The program reads in two input files:

```
parmn.txt
parmf.txt
```

The file `parmn.txt` simply contains the paths to the two input files. One the first line is the path to the master *A* file and on the second line is the path to the moderate *B* file.

The file `parmf.txt` contains the parameter information that controls the program execution. The first line of the file contains 10 integers:

1. The number of blocking criteria or blocking strategies
2. The number of matching fields
3. The number of sequence fields
4. The output cutoff flag
5. The number of ID fields
6. The print output flag
7. The record duplicate flag
8. The number of *B* file records
9. The length of an *A* file record
10. The length of a *B* file record

The program allows the user to test the *A* file against several blocking strategies simultaneously. The first number tells the program how many blocking strategies are to be used. The current maximum number of blocking strategies is 10.

The matching fields are the fields that the program will compare to compute a matching weight for a pair of records from files *A* and *B*.

The sequence fields are the fields that contain the record identification number for each record in its respective file. Here the user indicates the number of fields that make up the sequence number.

A positive output cutoff flag indicates that the user intends to supply cutoff values for the matching weights for the *A* subfile output. A zero indicates that the program should use the default values, which are currently 16 and 0.

The ID fields refer to a unique record identifier such as the Census identification number. Here one indicates the number of fields that this ID number is spread over. Having the ID number is optional. If the user does not have the need of the ID number, this parameter value can be set to 0 and no ID field information will be used.

The print output flag is set to 1 if the user wishes to supply cutoff values for the matching weight of pairs of matched *A* and *B* records to be printed out for analysis purposes. If this flag is set to 0, the default cutoff values 12 and 0 are used.

The record duplicate flag is set to 1 if the user is trying to find duplicate records within a file or within a subfile of a master file. When this flag is set, the program checks whether the two records have the same sequence number (record identification number), and skips the matching comparison computation if they do. Thus when running a match on a file against itself, the program does not print out when a record is paired with itself. When not searching for duplicates within the same file, this flag should be set to 0.

If the number of records in the *B* file is known, it can be entered here. The number of *B* file records can be obtained from the report of the **blokstats** program, as described below. If the number of *B* file records is not known, then this input integer should be set to 0 and the program will count the number of *B* file records.

The *A* and *B* files are assumed to be flat ASCII files of fixed record length. In these last two positions, the user informs the program of the length of the *A* file record and *B* file record respectively.

The second line of the `parmf.txt` file is a row of integers providing the number of blocking fields for each blocking strategy. The number of integers on this line should equal the first number on the first line of the file.

The next lines of the `parmf.txt` file provide the parameters for the individual blocking fields. The number of these lines should equal the sum of the integers on the second line. Each line contains:

1. A blocking field name (up to 20 characters)
2. The start position in the *A* file of the blocking field
3. The length of the blocking field
4. The start position in the *B* file of the blocking field
5. The length of the blocking field
6. The blank filter flag.

The blank filter flag allows the user to specify whether the program should compute matching weights for pairs of *A* and *B* records that agree on their key value because both keys are blank. A blank key is one that consists entirely of either spaces or zeros. If the user wants the program to skip blank key comparisons for a particular blocking strategy, then the last entry in the blocking field should be a 1 for every matching field included in that blocking strategy. If any of the blank filter flags for a blocking field in a blocking strategy are set to 0, then pairs of records with blank keys will be compared.

The next lines of `parmf.txt` list the parameters for the matching fields. The number of these lines should equal the second integer on the first line of the file. These are the fields in the file records that are compared to compute a matching weight. Each line contains:

1. A matching field name (up to 20 characters)
2. The start position in the *A* file of the matching field
3. The length of the matching field
4. The start position in the *B* file of the matching field
5. The length of the matching field
6. A null flag (non-operational)
7. The field comparison type
8. The conditional matching agreement probability
9. The conditional non-matching agreement probability

For the field comparison type, one should choose among the following options:

- **c**—exact string comparison
- **uo**—string comparator allows for some typographical variation
- **p**—numerical comparison for age
- **y**—numerical comparison for year

For the agreement probabilities, one can use rule of thumb estimates. The first number represents the conditional probability that two records agree on the matching field value given that the two records represent a match. The second number represents the conditional probability that two records agree on the matching field value given that the two records do not represent a match. The agreement weight for this field value for two records is based upon the ratio of these two numbers. A larger ratio implies a stronger distinguishing power for that matching field. Presumably the ratio should always be larger than 1. In the full record linkage program, these conditional probabilities are computed from maximum likelihood methods based upon the distribution of the agreement patterns in the set of pairs of records. Since the purpose of the **bigmatch** program is to identify plausible matching candidates from file *A*, these values do not have to be estimated too accurately.

If the number of sequence fields was greater than 0 in the first line of `parmf.txt`, the next lines contain the parameters for the sequence number fields. The sequence numbers should be record identification numbers for locating records in the original *A* and *B* files. The number of these lines should equal the number of sequence fields indicated in the first line of `parmf.txt`. Each line should contain:

1. The sequence field name
2. The start position of the sequence field in the *A* file

3. The length of the sequence field
4. The start position of the sequence field in the *B* file
5. The length of the sequence field

If the output cutoff flag was set to 1 in the first line of `parmf.txt`, then the next line contains the upper and lower output cutoff values. Actually only the lower cutoff value is very important since the program outputs an *A* file record as a plausible match candidate if it finds a *B* record with an agreeing key which has a matching weight above the lower output cutoff value. Thus the size of the matching candidate output files can be adjusted by varying the lower output cutoff.

If the number of ID fields was set greater than zero, then the next lines contain the parameters for the ID fields. This section was designed for collecting the Census identification number for each file record. The number of these lines should equal the number of ID fields indicated on the first line of `parmf.txt`. The format for these lines is the same as that for the sequence fields.

Lastly if the print cutoff flag was set to 1 on the first line of `parmf.txt`, then the next line should contain the upper and lower cutoff values for the matching pair output files. If a record from the *A* file has a key value that agrees with the key of a *B* record, the matching weight of the record pair is computed. If the matching weight is between the upper and lower print cutoff values and between the upper and lower output cutoff values, then the matching information of the two records is printed to a print output file. The purpose of the print output files is to enable the user to analyze how the program is performing its matching comparison computations. By specifying values of upper and lower print cutoff within the bounds of the output cutoffs, one can reduce the amount of output and concentrate one's analysis one a smaller range of the matching weight comparison function.

Program Output

There are three types of output files from the **bigmatch** program. Corresponding to the *n*th blocking strategy, $n = 0,1,2,\ldots$, there are files

- *A* file subfiles `aout`*`n`*`.dat`
- Record pair matching field information
    - Positive matching weights `prnt1_`*`n`*`.dat`
    - Negative matching weights `prnt2_`*`n`*`.dat`
- Summary data `summ.dat`

For each blocking strategy *n*, the subfiles `aout`*`n`*`.dat` contain complete *A* file records that have a key that matches a key of a record in the *B* file where the record pair has a matching weight above the lower output cutoff value.

For the *n*th blocking strategy, the program prints out matching information for each record pair where the keys agree and the matching weight is between the lower and upper print cutoff values (and the lower and upper output cutoff values). The program prints out the matching weight, *A* file sequence number, *B* file sequence number, common key value, *A* record matching field values, and *B* file matching field values. Any such record pairs with positive matching weight get printed out to `prnt1_n.dat` and any with negative matching weight get printed out to `prnt2_n.dat`.

The file `summ.dat` contains summary information about the program run for diagnostic purposes. The file gives the total number of *A* and *B* records read. For each blocking strategy, it gives the number of distinct *B* keys and the number of *A* records output to the subfile. Then there is a table listing each *B* key, the number of *B* records with that key, and the number of *A* records with that key which were output to the subfile.

Reformatting Program Output

The output to files `prnt1_n.dat` and `prnt2_n.dat` can be hard to read since the records are long. These files can be reformatted into one file by running the script **printout** that calls the C program **printer**. The resulting output file:

- Is sorted by decreasing matching weight
- Has each set of record pair information printed on 3 lines
    - First line has matching weight, sequence numbers, and key value
    - Second line has *A* record matching field values
    - Third line has *B* record matching field values

These programs are used with the full matching program to aid the user in determining cutoff values for designated links and non-links. We should note that unlike the full matching program, the output of the **bigmatch** program does not necessarily have to be one-to-one. That is, the output file of record pairs may have *A* records paired with more than one *B* record and *B* records paired with more than one *A* record.

In order to use the program you need to:

1. Use *chmod* to make the script **printout** an executable
2. Compile the program **printer.c** using the C compiler, naming the executable **printer**

Once these programs are prepared, you can reformat the files from the *n*th blocking strategy by the following steps:

1. Copy the file `prnt1_n.dat` to the file `prntd1.dat`
2. Copy the file `prnt2_n.dat` to the file `prntd2.dat`
3. Execute the script **printout**

The output file is called `final.out`. You may want to rename it to `finaln.out` or something that indicates the number of the blocking strategy that it represents.

Evaluating Blocking Strategies

The time that the **bigmatch** program takes to run is most affected by the number of *B* file records that have the same key. Each time an *A* record is found with a given key, the matching weight is computed with every *B* record that has this key. A particularly inefficient case can occur when a blocking strategy produces a large number of *B* key records with a blank key. Every time an *A* record is found with a blank key, it will be compared with all of these *B* key records even though there is not much evidence of the records being similar. As a tool to help the user formulate more efficient blocking strategies, one can use the program **blokstats**.

The **blokstats** program reads in only the *B* file and the parameters for the *B* file blocking strategies. It prints out a report file that indicates the number of *B* file records per key value and the number of blank keys. The user can use this information to adjust the blocking strategies before reading the large *A* file.

The program `blokstats.c` can be compiled with a C compiler. On the UNIX machine, there is no need to link in the math library.

As with the **bigmatch** program, there are two input parameter files. One file is `parmn1.`txt, which contains one line which gives a path to the smaller *B* file.

The second input parameter file is `parmb.txt`, which gives a description of the *B* file key layouts. The first line has a sequence of 3 integers which provide the following information:

1. The number of sets of blocking criteria or blocking strategies—There is not a hard-coded upper limit for this number, but more blocking strategies require more memory.
2. The number *n* of desired most common *B* keys to be reported out
3. The total length of a record in the *B* file

The next line contains the number of blocking fields for each blocking strategy, so that the number of entries on this line equals the first number in the line above. The sum of the numbers on this line equals the number of blocking fields listed on the lines below.

The next lines list the blocking fields. They have the blocking field name, the starting position of the field in the *B* record, and the length of the blocking field.

The output file is `blkrpt.dat`. It lists the total number of *B* records read, then for each blocking strategy, it lists the number of distinct keys, the average number of *B* records per key, the number of records with blank keys (if any), and the *n* keys with the largest number of records, where *n* is the number specified in the parameter file above. A blank key is defined to be one that consists entirely of space characters or entirely of 0 characters.