AN OUTLINE OF IR'S ADHOC MATE-PAIR VERIFICATION PROCESS

INPUTS:

1. **`Input.labdata`**. We download from SCI_LDS a table with a row for each sequenced read and the following 7 columns:
   A. Fragment UID: 64-bit ID assigned by our unique ID system (FRAG),
   B. Library name: e.g. "DM4" through "DM12" for Drosophila (LIBN),
   C. Clone Plate #: unique integer for plate assigned by lab data system (CLNP),
   D. Clone Well #: well on clone plate, an integer in 1..384 (CWEL),
   E. Clone End: 5 and 3 for forward and reverse (CDIR),
   F. Sequencing Plate #: unique integer for plate assigned by lab data system (SEQP),
   G. Sequencing Well #: well on sequencing plate, an integer in 1..384 (SWEL).

2. **`Input.trimreads`**. A proto IO file containing fragment messages for every sequenced fragment. This file was produced from a download of PROD_IDS.

3. **`LibData`**. A small table, hand-crafted in our case, with a row for each library and the following 4 columns:
   1. Library name: e.g. "DM4" through "DM12" for Drosophila,
   2. LibraryUID: 64-bit ID assigned by our unique ID system,
   3. Average length: estimated length of clone inserts in base pairs,
   4. Standard deviation: estimated standard deviation of insert length distribution.

4. **`FullBACS.seq`**. A FASTA file of all finished BAC's for the organism being sequenced (95 finished BACs totalling 22Mbp for Dros, this will be more like 1000-1500 BACs totally 300Mbp for human).

5. **`DrosRepeats.seq`**. A FASTA file of all known repeats for the organism (obtained from Ming Lee for Dros, to be taken from the RepeatMasker library for human).

OBJECTIVE:

A proto-IO file of link (LNK) and distance (DST) messages where all suspect mate pairing relationships have been removed from consideration, where the mean and variance of each library is accurately estimated, and where library mistracking has been corrected if possible. In the current implementation 3 different files are output, although only one `D.links`, is ultimately feed to the assembler.

1. **`D.links`**. A proto IO file containing all the finally approved mate pair link messages and a distance message for each library.

2. **`D.rereads`**. A proto IO file containing all the confirmed reread pair link messages in the data set.

3. **`D.dupfrags`**. A table with a single column, each row containing the UID of a fragment that is an exact duplicate of one left in the dataset.

METHODOLOGY:

To do this quickly our implementation centers on "awk" scripts and a couple of C-coded utilities that manipulate what I call "awk tables". A table has a fixed number of columns/fields per row/line. The first line is always a label line that contains a 4-letter symbolic name for each column (e.g. FRAG for the fragment UID column, LIBN for the library name column, etc.) The remaining rows contain the data where each field is separated by a space character.

Several general utilities are (1) "DrosSorter" that sorts an awk table lexicographically on any subset of columns, (2) "DrosJoin" that does a relational join on the columns common to two awk tables, (3) "DrosProject" that reorders and outputs a selected set of columns of an awk table, and (4) "DrosSelect" that selects rows from an awk table that satisfy a given predicate.

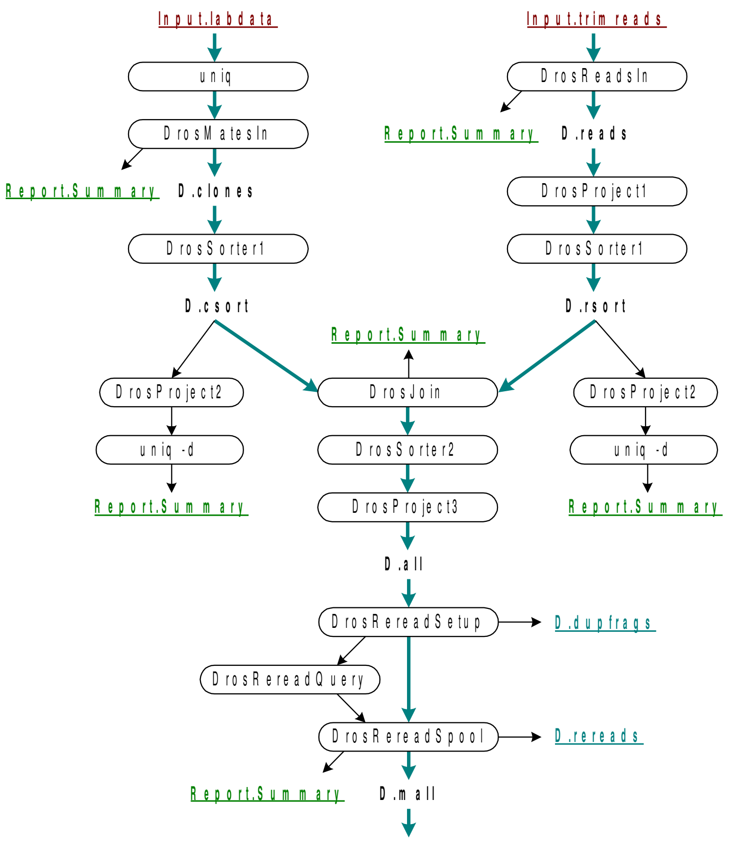
The remainder of the utilities are specific to the purpose at hand, most are implemented as awk scripts. The entire process is driven by a Makefile whose dependencies are established so that requesting the goal file to be made, sets off the portion of the entire process that needs to be performed to update it. *Many of the utilities contain code that checks preconditions for the data and also to analyze the data set in various ways (as I developed this process flow, I was continually evaluating the data.)*

PROCESS FLOW:

The following two diagrams shows the entire flow of data through the various programs in the process followed by a description of what each does. Input files are shown in brown and underlined, output files are shown in blue and underlined, a report summary file, "Report.Summary", that is appended to by various steps is shown in green, and the remaining files are intermediate files that are targets for the makefile that performs whatever processing is necessary to update the target. Process steps are shown within rounded boxes. The main flow of the per-fragment record data is shown in a thick blue line.

- **`uniq`**: Remove all duplicate lines in the input (this did occur in some early version of "Input.data").
- **`DrosMatesIn`**: Remove all rows for which either (a) the fragment UID is "NULL", or (b) the sequence well # doesn't equal the clone well # (some of these do occur in early data). Output the table but replace the CWEL and SWEL columns with a single WELL column containing the well number that is guaranteed to be the same in both columns.

`D.clones`: An awk table with the columns (FRAG,LIBN,CLNP,CDIR,SEQP,WELL).



- **`DrosReadsIn`**: Convert proto IO messages to an awk table with the following size columns: (1) the fragment UID (FRAG), (2) the entry time (TIME), (3) the clip interval beginning (CLPS), (4) the clip interval end (CLPF), (5) the sequence read (SEQN), and (6) the quality vector for the read (QLTY). Remove any message that (a) does not contain all six fields above, (b) has a clip interval out of range for its sequence, or (c) has a quality vector whose length differs from that of its sequence.

`D.reads`: An awk table with the columns (FRAG,TIME,CLPS,CLPF,SEQN,QLTY).

- **`DrosProject1`**: Remove the quality vector column (QLTY) from the input file (it is not needed by the current implementation of the mate pairing process).
- **`DrosSorter1`**: Sort the incoming files on their fragment UID column.

`D.csort`: Same fields as “D.clones” but now sorted on fragment UID.

`D.rsort`: An awk table with the columns (FRAG,TIME,CLPS,CLPF,SEQN) sorted on fragment UID.

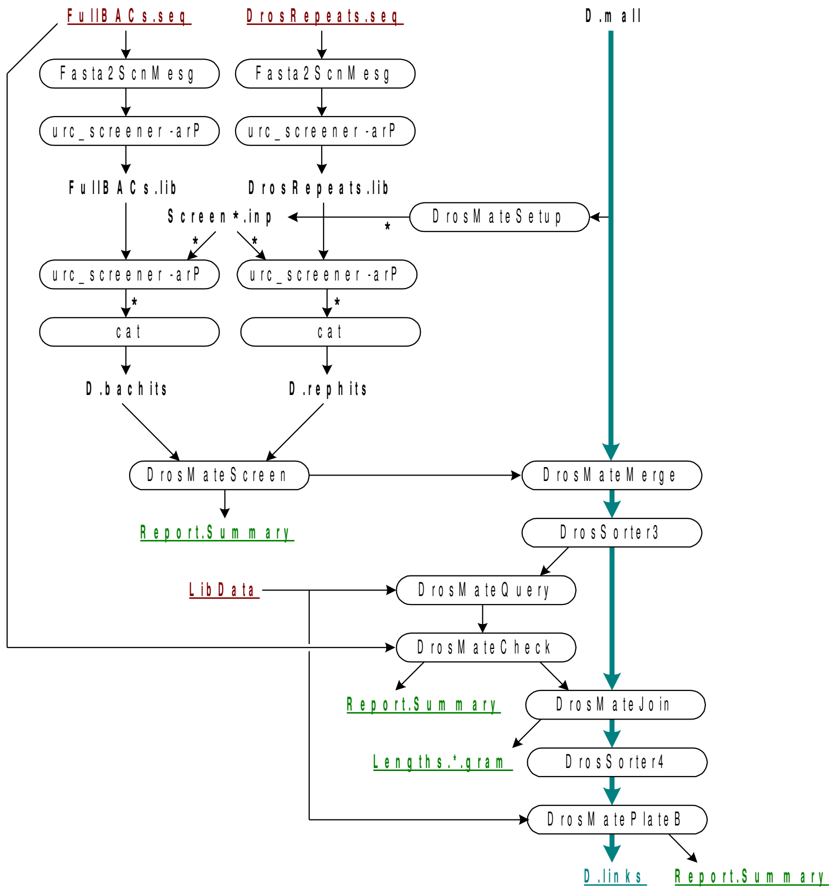
- **`DrosProject2`**: Output only the fragment UID (FRAG) column of the table and then …
- **`uniq -d`**: … complain if there are any duplicate UIDs.
- **`DrosJoin`**: Join the the tables “D.csort” and “D.rsort” on the one column they share in common and on which they are both sorted: FRAG, the fragment UID column. One now has records with the combined laband the sequencing data for each fragment.
- **`DrosSorter2`**: Sort the input in lexicographical order of clone plate (CLNP), clone end (CDIR), and lastly well number (WELL).
- **`DrosProject3`**: Reorder the columns of the incoming file to produce “D.all” below.

`D.all`: An awk table with the columns

(LIBN,CLNP,WELL,CDIR,SEQP,FRAG,TIME,CLPS,CLPF, SEQN) sorted lexicographically on (CLNP,CDIR,WELL).

- **`DrosRereadSetup`**: Because of the sorting on "D.all", rereads in this file will be in consecutive rows that have the same (CLNP,CDIR,WELL) values. For any pair of rows where it is further true that the sequences are identical, one of the two fragment UID's is output to "D.dupfrags" and eliminated from further processing. All remaining groupings are rereads and a special file containing queries for each possible reread pair is prepared for input to "DrosRereadQuery". Each query consists of a line with the clip ranges for the two sequences followed by the two sequences on the next line. All lines that did not get discarded are further output and shunted forward to "DrosRereadSpool" (see the diagram).
- **`DrosRereadQuery`**: For each query prepared by "DrosRereadSetup", attempt to find at least a 50bp overlap between the clear ranges of the two sequences allowing up to 15% differences. If find such an overlap output a line with the amount of overhang with respect to the 1st-sequence of the pair, otherwise output –10000, and impossible value for sequences whose length is 1024 or less.
- **`DrosRereadSpool`**: Simultaneously read the answers to the queries produced by "DrosRereadQuery" and the non-duplicate fragment records output by "DrosRereadSetup" so as to get the answer as to whether a given pair of rereads really do appear to be rereads as you go. For a given set of potential rereads, if there is any inconsistent overlaps in the set, then remove all the reads from consideration. If they are all consistent, then for each fragment except the one with the largest clear range, output a REREAD link message to the file "D.rereads" that links the fragment to the largest one and remove the read from further consideration. All other reads are output to "D.mall"

`D.mall`: An awk table with the same columns as "D.all" except that all duplicate reads, inconsistent reread sets, and all dominated rereads in consistent sets have been removed. It is now the case that each (CLNP,WELL,CDIR) triple from a given row is unique.



- **`Fasta2ScnMesg`**: Convert the FASTA file entries into screen item (SCN) proto IO messages to give to "urc_screener" (the Celera assembler screener module). *It should be noted that currently this step and the "urc_screener" call following it are performed manually and not coded in the Makefile.*
- **`urc_screener -afP`**: Given a set of screen item messages, this call produces a corresponding ".lib" file. Given a ".lib" file and a set of proto-IO fragment (IFG) message, it outputs a set of (SFG) fragment messages with screen matches included in the record.
- **`DrosMateSetup`**: Take "D.mall" and convert the rows for forward reads (CDIR == 5) that have a mate pair into proto-IO fragment (IFG) messages, with 75,000 into each of a sequential series of files "Screen001.inp", "Screen002.inp", …. Each of these files is screened against "DrosRepeats.lib" and "FullBACs.lib" as separate jobs*, currently done manually as an LSF submit.* This step obvious chews up a lot of time, hence the desire for parallelism. The results come out in a sequence of files "Screen001.urc", "Screen002.urc", that when cat'd back together produce "D.rephits" and "D.bachits", respective of which library the files where screened against.

`D.rephits`: For every forward read that has a mated reverse read in "D.mall", in the same order, a proto IO fragment message of type SFG with the hits against the Dros. repeat library (of length at least 40bp and 10%??? fidelity) contained therein.

`D.bachits`: Same as "D.rephits" except screened against the finished BACs.

- **`DrosMateScreen`**: Reads the fragment messages in both "D.rephits" and "D.bachits" in synchrony so that for each paired forward fragment one has available both its finished BAC hits and the repetitive element hits. For each message the fragment UID followed by a pair of integers is output on a line indicating whether a unique BAC hit was found and if so where on the BAC. The *repeat interval* of a fragment is the segment of the fragment from the first base (if any) that is is in a repeat match to the last base that is in a repeat match (not necessarily the same repeat match). A *unique BAC hit* exists for a fragment iff it has a match to only one BAC and at least 50 contiguous bases of this BAC match are not in the repeat interval. In this case the BAC # is output (#'d consecutively starting at 1) as the first integer followed by either (a) the location in the BAC of the 3' end of the fragment if aligned in the forward direction, or (b) the negation of the the location in the BAC of the 3' end of the fragment if aligned in the reverse direction. If a unique BAC hit does not exist then either the pair "0 –2" is output if there was a unique hit to a BAC but it was nullified by matches to repetitive elements, and "–x –1" is output where x is 0 if no BAC matches were found or the numbef of distinct BAC matches if 2 or more were found. The output is thus a list of triples of numbers of the same length as "D.mall", one "answer" per fragment record.

- **`DrosMateMerge`**: The rows of "D.mall" and the unique BAC hit results for its paired, forward reads are read in synchrony. Two fields are added to each record, BACN and OFFS, corresponding to the unique BAC hit and location computed by "DrosMateScreen". For paired, forward read records, the integer pair for the fragment is added. For paired, reverse read records, the integer pair for its forward mate is added. For all other fragments "0 0" is output. One now has records where the unique BAC hits for each paired, forward read are attached.
- **`DrosSorter3`**: Sort the result of "DrosMateMerge" lexicographically on BACN (unique BAC hit for the read), CLNP, WELL, and CDIR. Thus all reads hitting a given BAC are contiguous and mates are also consecutive.
- **`DrosMateQuery`**: Prepare a query file for "DrosMateCheck" below, that for each mate pair whose forward read has a unique BAC hit, contains a line containing BAC #, location of the forward hit on the BAC, estimated length of the insert for the pair, and clip ranges for the forward and reverse reads, followed by a line containing the forward read and a line containing the reverse read. Again note that these queries are ordered by BAC #.
- **`DrosMateCheck`**: Given the list of queries produced by "DrosMateQuery", determine which pairs are confirmed by the BAC, which are refuted by the BAC, and which are neither confirmed or refuted (for one of several reasons to be described below). Let A be the clear-range trimmed forward read anb B be the clear-range trimmed reverse read of a query pair. The goal is to output an integer code indicating the outcome followed in the case of confirmation by the distance between the mates in the BAC. If the pair is for a 2Kbp insert then we look for B within 15Kbp of A, and if the pair is for a 10Kbp insert then we look for B within 25Kbp of A. Given the position of A's unique match to the BAC, if B is less than 165bp in length or could possibly be located off the end of the BAC then "-1 0" is returned to indicate that the A,B pair cannot be confirmed or refuted. Next, we look for a complete match between A and its BAC permitting 15% differences. "-3 0" is returned if A's match to the BAC is not end-to-end at this level of similarity. Furthermore if the A-match is not at the 2% or better level then "-2 0" is returned. We are thus interested only in that status of pairs where the A-match is unique to the BAC, end-to-end, and of high fidelity. We then try to match B to the BAC in the appropriate range relative to A. A very loose match criterion is applied: a $\leq$15% match of a 100bp segment of B beginning at positions 15, 65, 115, … 50k+15 where $k = \lfloor |B|-165 / 50 \rfloor$. The idea is to trend towards confirming rather than refuting. If such a match cannot be found then "0 0" is returned, otherwise "1 x" is returned where x is the distance between the 5' ends of A and B as they position themselves against the confirming BAC. The list of these results are output and will be joined with the main stream file output by "DrosSorter3".
- **`DrosMateJoin`**: Joing the query results from "DrosMateCheck" with the main stream of fragment records forwarded from DrosSorter3. Append the two query integers to each fragment record adding them as a PAIR and DIST column. For those mate pairs whose forward read did not have a unique BAC match append the pair "-4 0" and for those forward reads that were not paired append the pair "-5 0". In summary, the PAIR column contains a "pairing code" with the following values and interpretations: (-5) unpaired forward read, (-4) read in a mate pair whose forward read did not have a unique BAC match, (-3) read in a mate pair whose forward read did not have an end-to-end match at its

unique BAC location, (-2) read in a mate pair whose forward read had more than 2% error in its end-to-end match at its unique BAC location, (-1) read in a mate pair whose reverse read may be off the end of the BAC at which the forward read was uniquely located, (0) a read in a mate pair refuted by the BAC library, and (1) a read in a mate pair confirmed by the BAC library. In the case the PAIR column is 1, then the DIST column contains the distance between the mates. The negative codes are basically all diagnostic for purposes of analysis. The essential thing is either a mate pair is refuted, confirmed, or neither refuted nor confirmed. In addition to the new columns, this utility also outputs files with the names "Length.<libname>.gram", one for each clone library. The file contains a list of the confirmed distances between mate pairs in a given library and can be passed to "celagram" to be viewed as a histogram. *Viewing this histograms permits one to refine the clone length average and standard deviation, this was done by manually editing the "LibData" file before executing the final step "DrosMatePlateB". Viewing these histogram also reveals potentially library mislabellings seen a twin peaks.*

- **`DrosSorter4`**: Sort the input lexicographically on (CLNP,WELL,CDIR) to get plates contiguous in the data set.
- **`DrosMatePlateB`**: For each pair of forward and reverse plates, we determine if the plate pairing is verifiable as a good paiir. The basic criterion is that there be a total of more than 10 pairs that are confirmed or refuted and that there are 4 times more confirmed pairs than refuted pairs. In addition, the average distance of the confirmed pairs must be inside the interval defined by the expected mean for the library plus or minus 3 standard deviations. In addition, before determining if a plate pair is verifiable, we first attempted to correct library mistracking (e.g. a colony plate was labelled DM5 when it was actually a DM7 plate) as follows. If a plate has more than 5 confirmed pairs, then if the average distance of the confirmed pairs is outside the expected mean plus or minus 3 standard deviations, then we checked the following alternatives: DM7 could be DM5, DM5 or DM9 could be DM7. These corrections were hand tailored based on knowledge of the time frames in which these libraries were sequenced. If the distance interval for the library correction alternative contains the average distance, then the correction is made. *Normally this correction should be reflected back into the fragment database but was not as part of this ad hoc procedure.* ("DrosMatePlateA" was an earlier version of this last phase in which I was investigating the nature of the confirm/refute data.)