BIOINFORMATICS

Vol. 17 no. 12 2001
Pages 1093–1104

DNA sequence quality trimming and vector removal

Hui-Hsien Chou[1] *and Michael H. Holmes*[2]

[1]*Department of Zoology and Genetics, Department of Computer Science, Iowa State University, Ames, IA 50011, USA and* [2]*Department of Bioinformatics, The Institute for Genomic Research, 9712 Medical Center Drive, Rockville, MD 20850, USA*

Received on March 8, 2001; revised on June 11, 2001; accepted on June 12, 2001

ABSTRACT

Motivation: Most sequence comparison methods assume that the data being compared are trustworthy, but this is not the case with raw DNA sequences obtained from automatic sequencing machines. Nevertheless, sequence comparisons need to be done on them in order to remove vector splice sites and contaminants. This step is necessary before other genomic data processing stages can be carried out, such as fragment assembly or EST clustering. A specialized tool is therefore needed to solve this apparent dilemma.

Results: We have designed and implemented a program that specifically addresses the problem. This program, called LUCY, has been in use since 1998 at The Institute for Genomic Research (TIGR). During this period, many rounds of experience-driven modifications were made to LUCY to improve its accuracy and its ability to deal with extremely difficult input cases. We believe we have finally obtained a useful program which strikes a delicate balance among the many issues involved in the raw sequence cleaning problem, and we wish to share it with the research community.

Availability: LUCY is available directly from TIGR (http://www.tigr.org/softlab). Academic users can download LUCY after accepting a free academic use license. Business users may need to pay a license fee to use LUCY for commercial purposes.

Contact: Questions regarding the quality assessment module of LUCY should be directed to Michael Holmes (mholmes@tigr.org). Questions regarding other aspects of LUCY should be directed to Hui-Hsien Chou (hhchou@iastate.edu).

INTRODUCTION

There have been a lot of studies on how to align DNA sequences with each other. Most of these studies are based on a fundamental assumption that the sequences being compared are trustworthy, i.e. their bases are correct. However, in large genome sequencing centers, the raw DNA data coming directly from automatic sequencing machines often violate that assumption. In fact, most raw DNA sequences have base-call errors. Nevertheless, to make good use of these unreliable raw sequences, we still need to compare them against some other data, such as vector and contaminant sequences, in order to prepare them for later stages of the genomic data processing pipeline (e.g. fragment assembly or EST clustering).

This creates a dilemma. We have uncertain raw DNA sequences, but we need to compare them against other sequences in order to obtain usable data. This data cleaning stage is critical for the success of later stages of the genomic data processing pipeline, as illustrated in Figure 1. Through this cleaning stage, unreliable raw data from sequencing machines can be enhanced to a more reliable level, so that later stages of the processing can use them without concern about their *base* quality[†]. Our solution at The Institute for Genomic Research (TIGR) for this cleaning stage is a specialized tool that was designed to handle this stage of genomic data processing. After more than two years of practical use in TIGR and several rounds of improvement, we believe it is mature enough now to be introduced to the scientific community. We hope other researchers can also benefit from using it.

METHODS

This tool, called LUCY, was designed to take the base-call quality assessment of each base into consideration in the cleaning process, to make sure the processed sequences have the best *overall* quality possible based on their *individual* base quality values. Thanks to a few powerful chromatogram base-calling programs like *phred* (Ewing *et al.*, 1998; Ewing and Green, 1998) and *TraceTuner* (Paracel, 2000), the quality value for each individual base of a raw sequence can be estimated. The individual base quality assessment is generally reliable, but a sequence can have very different base qualities among its bases.

[†] However, other quality issues may still exist, such as the need to remove chimeric sequences at the assembly stage.

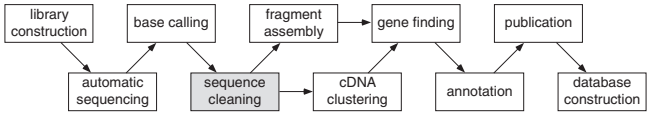

Fig. 1. Major steps of a typical genomic data processing pipeline. LUCY is designed to handle the *sequence cleaning* stage between the base calling and fragment assembly or clustering stages.

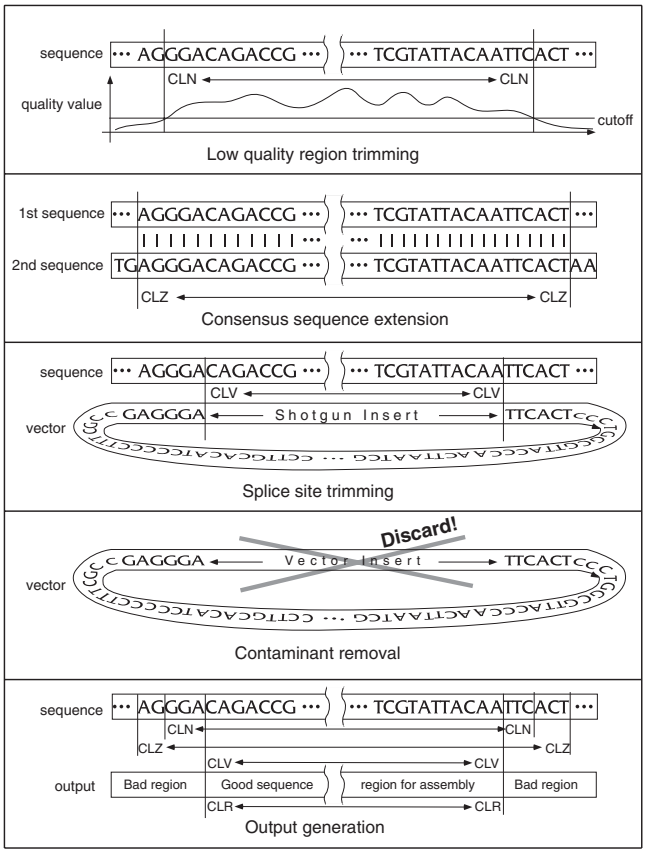

Fig. 2. Illustrations of LUCY's major processing steps. See the main text for explanations.

LUCY's task is to identify the largest subsequence that is of sufficiently high quality and also free of contaminating vector sequence.

LUCY does its job by first determining the longest continuous high quality region of a sequence based on its base quality values. Some bases within this *good quality region* may be wrong, but the overall value must be greater than a user-specified minimum value (97.5%)[†]. If a second base-call sequence can be obtained using a different base-calling program with the same chromatogram file, LUCY may use it to extend the good quality region by aligning the second sequence with the first one at places where they match with high fidelity. The rationale is that if two independent base-calling algorithms agree on some bases, chances are these bases are correct even though some of them may have lower quality values. This method works well in processing *ABI 377* and *370* chromatograms where the second sequence comes from the *ABI base caller* and the first sequence comes from *phred*. This method does not work very well on *ABI 3700* sequences, probably because the new base-caller on the *ABI 3700* uses an algorithm similar to that used by *phred*. Generally, the step to extend the good quality region can be skipped without sacrificing too much useful data, but potentially it may provide that extra linking leg needed to connect two contigs together during fragment assembly.

After having determined a good quality region, LUCY then searches for vector splice sites on the sequence, and makes sure that they are not included in the usable region of the sequence. This step is especially difficult because most of the vector fragments lie in the low quality region of a sequence, so their existence and true boundary cannot be easily determined by straightforward sequence alignment methods. What LUCY does to solve this difficulty is to search for the splice sites first at a fixed region at the beginning of each sequence, taking into consideration the low quality of the base calls in this region. If that attempt fails, LUCY will search again *adaptively*, taking into account the boundaries of the good quality region determined earlier. When searching for splice sites, LUCY applies different search stringencies at different areas of a sequence to cope with the variable quality of base calls. LUCY also considers many special cases that were found through experience at TIGR. Many incremental improvements have been made to this step since the early versions of LUCY due to the difficult nature of this problem. We believe that LUCY now functions very well with most input data.

Following the splice site trimming step, contaminant sequences can be quickly identified and removed from the input data. Because all vector splice sites have been trimmed during the previous step, the contaminant removal step becomes relatively easy. Note that we are concerned only with the quick *detection* of contaminants, not the *details* of the contaminants. Finally, markers identifying the usable region of each sequence are generated in the output. Illustrations of LUCY's major processing steps are given in Figure 2. LUCY's processing steps are generally independent of each other and they produce independent trimming pairs using the `CL?` tags shown in

[†] In this paper, we include the default value of a user-adjustable parameter in parentheses.

the figure[‡]. When we refer to the *good region* in this paper, we mean the current cumulative trimming result after each processing step, i.e. the narrowest region obtained by combining all trimming markers. At the end of LUCY processing, the useful region of a sequence becomes the CLR region as shown in the figure.

Most of LUCY's parameters can be set by the users using its command line arguments. In the following discussion of each processing step, we represent parameters that influence a specific step in the `typewriter font`. We also enclose their default values in the parentheses after the parameter names. Generally, these default values were determined based on our past experience. Users can change these settings if they need to adjust LUCY's behavior in their own data environment, but it should be done with caution. In the user document distributed with LUCY, we provided additional information to help users determine some parameters for their specific environments.

Except for the quality region determination step, all other LUCY steps described below can be skipped if users do not supply the necessary input data for a step.

Quality region determination

The goal of this step of LUCY's processing is to find the longest region of each input sequence whose base calls are of sufficiently high quality to warrant using them. We will refer to this region as the *clean range* in this section. This step makes use of the data in the quality files that are provided as input. The quality files are created by the base-calling software before LUCY is run.

The raw data files (chromatogram files) from automated DNA sequencing machines, such as the *ABI 377* and *3700* sequencers, contain four sets of data traces, one each for the A, C, G and T sequencing reactions. The temporal relationship among the four traces is known, because the four reactions are run through the sequencer at the same time, in a single capillary or a single lane of a polyacrylamide gel. The traces can be visualized as a set of peaks, as shown in Figure 3.

It is the job of the base-calling software to examine the traces and determine the sequence of base calls. In addition to calling the bases, the base-calling programs *phred* and *TraceTuner* provide a quality value for each base called. The quality value is based on the estimated probability that the base call is in error. Quality (Q) and estimated base call error are related by the following formula:

$$Q = -10 \times \log_{10}(\text{probability of error}).$$

The base-calling programs base their estimates of the probability of error on a number of factors, including peak shape, spacing between peaks, signal strength and background noise (Ewing *et al.*, 1998; Ewing and Green, 1998). LUCY first converts the data from qualities to probabilities of error according to the formula above (e.g. $Q = 0$ is converted to probability of error $= 1.0$). All of the major steps in the quality trimming process involve calculating some *average* probabilities of error within certain *windows* along the length of the sequence. These steps are illustrated in Figure 4.

Since the beginning and end of each sequence are typically of low quality, the first step is to remove the lowest quality data from each end. This step is controlled by the `bracket` parameter. Starting from the left end of the sequence, LUCY finds the first window of size `window_size` (10) that has an average probability of error of `max_avg_error` (0.02) or less. Similarly, starting from the right end, LUCY finds the last window of size `window_size` meeting the same criterion. These windows and the bases between them are then subjected to the remaining steps, and the bases at the ends of the

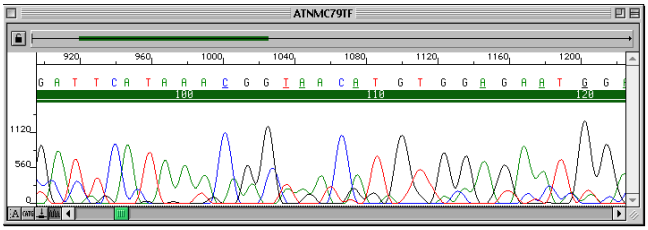

Fig. 3. The raw data in a chromatogram file can be viewed as four sets of overlapping peaks, one each for the A, C, G and T sequencing reactions.

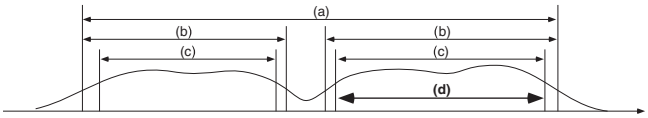

Fig. 4. LUCY's quality trimming steps. (a) Low quality areas are trimmed from each end, then (b) regions of poor quality within the sequence are identified and removed from the clean ranges. (c) The resulting candidate clean ranges are further trimmed to satisfy the overall average probability of error criterion and the criterion of the probability of error at terminal bases. (d) The largest remaining candidate is chosen as the final clean range.

[‡] These are historical TIGR database field names. Their meanings are CLN: clear of bad quality data, CLZ: clear of zgrasta (an alignment tool used to align the two base-call sequences before LUCY was available), CLV: clear of vector fragments, and CLR: clear of all bad things. The contaminant removal step does not produce trimming tags but sets the trash tag of contaminated sequences, which are represented by CLR left and right being both 0.

sequence that failed this step are excluded from further consideration.

The next step is to identify regions of the sequence that have unacceptably high error rates, and exclude those regions from the final clean range. This step is controlled by a set of `window_size` and `max_avg_err` pairs given to the `window` parameter[†]. By default, LUCY uses two window sizes in this step: a 50-base `window_size1` allowing a `max_avg_err1` probability of 0.08, and a 10-base `window_size2` allowing a `max_avg_err2` probability of 0.3. The purpose of the larger `window_size1` is to exclude large regions of poor quality, but it may fail to exclude smaller regions of very low quality. The smaller `window_size2` excludes those smaller regions.

LUCY starts with the candidate clean range from the first quality trimming step, and subjects it to the largest window from the `window` options. From the beginning of the candidate clean range, LUCY calculates the average probability of error in each window of size `window_size1`. Each such window that has an average error of `max_avg_err1` or less is added to a new candidate clean range, which keeps growing until some window fails this criterion, then the current candidate clean range is terminated. LUCY continues with the next window, and begins a new candidate clean range with the first subsequent window that passes the same error criterion. So this step starts with a single candidate clean range, but may result in several candidate ranges being produced.

Each of the candidate clean ranges produced by considering the largest window size is then subjected to the next window size (`window_size2`) in the same way, and so on, until all `window` sizes have been considered. Each of these steps may produce further fragmentation of the candidate clean ranges. However, any candidate clean range that is smaller than the length specified by the overall `minimum` parameter (100) is eliminated from further consideration, and in practice the number of sequences which produce multiple candidates of sufficient size is small.

The final quality trimming step is controlled by the `error` option, with `max_avg_error`[‡] and `max_error_at_ends` parameters. The `max_avg_error` parameter specifies the maximum overall average probability of error allowable in the final clean range (0.025), and the `max_error_at_ends` parameter specifies the maximum probability of error for each of the two bases at the ends of the clean range (0.02). For each of the candidate clean ranges produced by the preceding step, LUCY finds the largest subsequence that satisfies both of these criteria, and the largest such subsequence found among all the candidates becomes the final clean range.

[†] Do not confuse these with similar options associated with the `bracket` parameter; they are independent of each other.

[‡] Again, the `max_avg_error` here is independent to the other similar options associated with the other parameters.

In a small minority of cases, quality trimming may produce more than one candidate clean range that satisfies all of the specified criteria. Only the largest is kept, however. This point is discussed further in the Section **Discussion**.

Consensus sequence extension

After good quality region determination based on the quality of individual bases of a sequence, an optional step is to align the sequence to a second sequence, if available, that is obtained using a *different* base-calling algorithm with the *same* chromatogram file. The rationale is that if two different base-calling algorithms agree on a specific base call, then that base call is likely to be correct even if its quality confidence value is low. This is an attempt to extend the good quality region by using the result from a second base-calling program to boost the confidence level assigned by the first program. An important requirement when doing this step is that the second sequence must be obtained from a *different* algorithm. Otherwise, the second sequence can be almost the same as the first sequence and a false good region can be extended all the way to both ends of a sequence. This certainly ruins the purpose of the previous quality trimming step.

In TIGR, the first sequence is obtained from *phred* and the second sequence comes from the built-in base-calling program with the *ABI 377* sequencers. Normally, in good quality regions, the alignment is almost perfect between the two sequences. But in the lower quality areas, the *ABI base-caller* tends to call the *N* base (unknown) while the *phred* program almost always attempts to guess one. Therefore, this extension step works reasonably well due to the different characteristics of the base-callers. Recently we have stopped doing this extension step at TIGR. The reason for this is that we want to be able to state our quality trimming criteria in terms of base calling error probabilities. The sequence extension step makes this impossible. Another reason is that the new base-calling software bundled with *ABI 3700* seems to utilize similar base-calling algorithms as in *phred*, thus the sequences it generates are very similar to the phred output and cause this extension step to extend too much into the bad quality region. However, we retain this sequence extension feature in LUCY because it is still conceptually sound, and for those who are using *ABI 377* seuqencers, this step can sometimes extend the usable sequence length.

Since both sequences should be very similar in their good quality regions, instead of running a time-consuming dynamic programming alignment, LUCY attempts to quickly locate the *band of alignment* by doing the following. It converts the first sequence into 16 bp tags, sorts the tags and removes duplicates from the set. Each tag has an accompanying index linking back to its

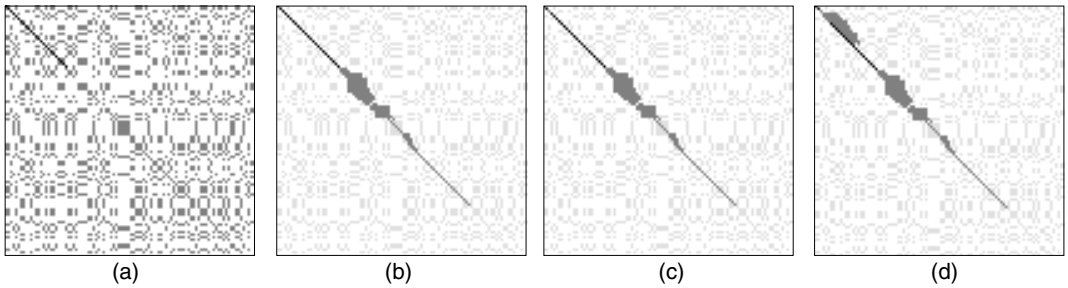

Fig. 5. Quick affinity region finding using the depth-first search algorithm in LUCY. (a) A partial dot graph showing matched bases between two sequences at the right end of an alignment. This partial dot graph is chosen such that the alignment band is right on the diagonal. (b) The search space (dark grey areas) LUCY explored when conducting the extension effort. The dot graph has been lightened in this and subsequent sub-figures to facilitate reading. (c) The LUCY search space after a mismatch has been introduced at the third base position in the figure. (d) The LUCY search space after a deletion is made to the third base position on one sequence.

position in the sequence[§]. Next, the second sequence is sequentially converted into 16 bp tags and searches are quickly conducted in the first sequence tag set to find the same tags. If some tags can be found on both sequences, their indices in the first sequence are compared against their indices in the second sequence, and their *differences* are recorded in a separate hit pool with associated hit counters. If a difference value already exists in the hit pool, its hit counter will simply be increased by one. At the end of the tag comparison processing the difference value with the highest hit count determines the relative offset of the two sequences in their best alignment regions.

Once the relative offset is known, their central alignment band (supposedly the good quality region) can be quickly located. Extension attempts are then carried out on both ends of this alignment band. Again, since the two sequences should be mostly similar except in the low quality regions, we only need to quickly connect the high affinity regions; we are not concerned with an optimal alignment between the two sequences in the low quality regions. Therefore, this can be formulated as a search problem using the more efficient depth-first search algorithm to locate the high affinity regions extending from the ends of the central alignment band. At every mismatched base between the two sequences on the current alignment band, there are three choices to continue the extension effort: skip one base on the first sequence, skip one base on the second sequence, or skip both. A stack is set up to record the choices being made at each mismatch position so the algorithm can come back to it later and try another choice. The depth-first search algorithm continues to find more high affinity regions and makes more choices at each mismatch until one of the following stop conditions becomes true: there are sufficient matched bases along the current search path such that the previous mismatch can be tolerated; the end of either sequence has been reached; there are over five mismatches along the current search path; or over 100 bases from the first mismatch has been scanned without finding sufficient matched bases to compensate the mismatches. These parameters come from the choice of an alignment stringency of 95%. Thus for every 100 bases, there can be no more than five mismatched bases.

Figure 5a depicts the *dot graph* around the right end of the alignment extension between two base-call sequences of a read[†]. The dark line at the upper 1/3 of the diagonal marks the end of the good quality region LUCY reports. Figure 5b reveals the actual search space LUCY explored when doing the alignment extension. We can see that LUCY's exploration is very confined and almost linear in the direction of the alignment diagonal. To test the search behavior of LUCY under different conditions, we intentionally create a mismatch pair at the third base of the portion of sequences covered in these dot graphs. It does not change the search space at all, as shown in Figure 5c. Because there are more than 20 matches beyond this mismatch, it is tolerated by LUCY and the end of the good quality region (the dark line) is not changed. Finally, we delete one base at the same test position from the second sequence to create a *gap*[‡] in the alignment. This deletion *smears* the LUCY search space a bit near the interruption. However, LUCY quickly recovers the rest of the high affinity region and reports the same end of good quality region as shown in Figure 5d.

The benefit of the depth-first search approach over dynamic programming alignment is in its almost linear-

[§] Duplicate tags at different locations may be lost, but this will not influence the outcome.

[†] Actual data from the right end sequencing of the clone ATIEP78 in the *Arabidopsis* project (Lin *et al.*, 1999).

[‡] No gap will actually be created in the output since only the good region on the first sequence will be used.

time speed. The disadvantage of this method is that it does not guarantee finding the longest alignment possible between the two sequences under the same alignment conditions. Since the search range is limited, when a much longer bad quality region separates two otherwise good alignments between the two sequences, this method will not be able to find the other alignment. Normally we will not use unconnected shorter good quality regions, so this limitation is not critical and is rare to occur. See the Section **Discussion** for more explanation.

Vector splice site trimming

LUCY requires a `vector_sequence_file` and a `splice_site_file` to conduct the splice site trimming and contaminant removal steps. The `vector_sequence_file` contains a single long sequence of the whole vector, which is used only in the quick contaminant removal step discussed next. The `splice_site_file` is what is actually used for the splice site trimming step. It contains two splice site template sequences upstream and downstream from the insertion point on the vector (see Figure 6). The splice site sequences are usually 100–150 bases in length, with some bases overlapping each other around the insertion point. They should also include any short linker sequences used during cloning reactions. Their actual lengths are not critical, as long as the splice site is totally covered.

With two splice site sequences in the `splice_site_file`, LUCY assumes all input sequences are read in the same direction as the splice site sequences. Usually users know the exact direction of their sequencing reactions, i.e. from which end of the clones they are reading the data, thus trimming needs to be done just along that direction. However, if that is not the case and the input may consist of sequences from both forward and reverse reads of clones (probably due to laboratory or sequence tracking errors), then LUCY can be instructed to do bidirectional trimming as well. If LUCY sees both forward and reverse splice site template pairs in the `splice_site_file` (i.e. a total of four splice site sequences), it assumes that bidirectional trimming has been prescribed. In TIGR, we always run bidirectional trimming despite the fact that only one of the trimming directions is actually needed. Trimming in the unnecessary direction might cause a few sequences to be shortened a little bit, but it guarantees that there can be no vector fragments in the good region even when the assumed read direction of some input sequences is wrong.

Because vector splice sites are usually at the beginning of a sequence where the quality of bases is low, a simple sequence comparison that looks only for the longest alignment with the upstream splice site sequence does not guarantee finding all vector fragments that may have been obscured by base-call errors. Instead, LUCY is programmed to find and remove all vector fragments that satisfy the search criterion. The quality of the base calls is usually poor at the beginning of a sequence, but gradually improves when moving into the sequence. Therefore, the search criterion has to be made *adaptive* to the average quality of the bases, such that in the low quality region we allow shorter vector fragments to be identified, but in the high quality region we require a more definite identification to avoid cutting good sequences short. Although there can be many different quality areas, we have found that using just three with low, medium and high-stringency search criteria serves the purpose well (see Figure 6).

The default `range` for the three search areas are 40, 60 and 100 bp, with three different minimum `alignment` lengths of 8, 12 and 16 bp. A local optimal alignment within each area must be equal to or longer than these minimum lengths before it is considered an identified vector fragment. This means that, by default, LUCY will search for splice sites in the first 200 DNA bases. If the splice site is not found in the first 200 bases or there are some special conditions that occur, additional searches beyond the default areas will also be conducted. When comparing sequences to the upstream splice site template, LUCY will first find the longest alignment with at least three matched bases for every one mismatch in it. This does not mean the alignment will have 25% errors in it because only the highest scoring local alignment is used. Shorter alignments to the left of the found one can be ignored. However, if there are other qualified local alignments that can be found after the best alignment, LUCY will continue the process until all potential vector fragments have been eliminated. After the end of the upstream splice site has been determined, the rest of the sequence will be searched for the downstream splice site with the highest (16 bp) stringency. This is to guard against short inserts.

The three initial search areas are fixed at the beginning of each sequence, without regard to where the good quality region begins. There may be questions as why they are not made to *stick* with the end of the low quality region (i.e. the beginning of the good quality region) so the `alignment` criteria are more in line with the actual quality composition of each input sequence. Actually, one previous version of LUCY was designed exactly that way, but that design was dropped later in favor of the fixed search areas for the *first* search attempt. There are several reasons for this decision. First, the average position of the splice sites is related to the specific vector being used and is more or less fixed at the beginning of each sequence, meaning that it depends much less on the sequencing quality. Therefore, shifting the search areas to the beginning of the good quality region often causes vector fragments to be missed for sequences that have

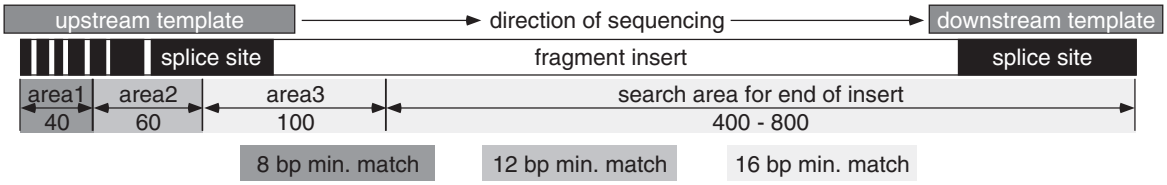

Fig. 6. LUCY's vector fragment searching areas on a typical sequence.

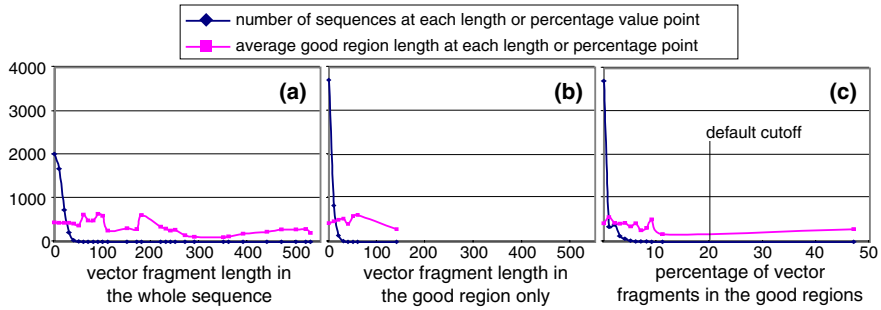

Fig. 7. Performance of the simple tag matching method in separating contaminants and good sequences when running against different regions of a sequence. (a) When running on the whole sequence, there is no clear separation between good and contaminated sequences. (b) When running only on the good region of each sequence determined during previous steps, a separation becomes possible since most sequences have far fewer contaminant matches in their good regions. (c) An even better separation can be achieved when taking the *fraction* of vector matches in the good region of each sequence. The default separation cutoff is set at 20%.

a longer initial low quality region. Although it can be argued that missing vector fragments in the initial low quality region is acceptable because this region will be excluded off the good region anyway, we find that it is still worthwhile to positively identify the position of vector splice sites for purposes such as estimating clone lengths, sequencing quality assessments, etc. Note that LUCY may conduct additional vector fragment searches under various special conditions. These additional searches are indeed adaptive to the good quality region. We will explain in the Section **Discussion** some of the special conditions that trigger additional searches.

Poly-A/T tail removal

If the raw DNA sequences are obtained from an EST library, some users want their poly-A/T tags to be removed before clustering. LUCY does this quickly after vector trimming by searching for the first `min_span` (10) or longer poly-T fragment within the first `initial_search_range` (50) bases inside the vector-free good region, then attempts to extend from this initial poly-T *seed* toward the center of the sequence, allowing no more than `max_error` (3) mismatches between every `min_span` (10) consecutive T bases in the scan. This is therefore a linear-time and linear-space operation. The poly-A tail trimming at the other end of the sequence is carried out similarly. If users wish to tell LUCY that they are processing EST sequences but they also wish to keep the poly-A/T tags for their purposes, they can issue the `keep` option in combination with the poly-A/T trimming option `cdna`.

Contaminant detection

Contaminants in the input sequences can come from many sources, such as *Escherichia coli* or human. However, most common ones are cloning vectors themselves. A vector can potentially splice with another vector, forming a *vector insert*. A vector can also pick up a very *short insert* that causes much of the sequence read to be the vector itself. These events do not happen frequently, roughly at the order of one in a few thousand sequences, but they do happen. The actual frequency depends on the specific vector being used for cloning and the laboratory processing. Since these are rare events, it is not worthwhile to spend a lot of time screening for a few contaminants. We want a method that can quickly identify potential contaminants and throw them off. We do not need to know where in the contaminated sequences the contaminants match.

Our solution is to pre-construct a tag pool from the full-length `vector_sequence_file`. The tag pool is made of every `vector_tag_size` (10 bp) fragment of the

contaminant sequence. Tags for the reverse strand of the contaminant are automatically generated and put into the same pool as well. The pool is then sorted and duplicate tags are removed. This is similar to the good quality region extension step mentioned previously, but it needs to be done only once for the entire running session of LUCY. During screening, each input sequence is converted into tags and searched against the contaminant tag pool to see if there is any match. We count the number of matches found on a sequence and use that to determine if the sequence is a contaminant.

This method may sound too simple to be effective. Indeed, if we try it on a test set of 5022 sequences from the *Arabidopsis thaliana* genome project[†] (Lin *et al.*, 1999) as shown in Figure 7a, it does not seem to tell us exactly which hits are real contaminants. Most sequences (98%) have less than 30 bp total match to the contaminant sequence. However, beyond 30 bp, the distribution seems to go uniformly all the way up to 530 bp. Two percent of the test sequences have nontrivial good regions and should not be simply discarded. Most of them are just shorter insert sequences that still contain valuable data. We need a better way to distinguish useless sequences from useful ones.

Thanks to the sophisticated vector splice site trimming step above, if we run this contaminant search only within the *good region* of each sequence, the distinction becomes more obvious. Basically, this rules out all vector splice sites in the comparison and therefore reduces matches to the contaminant found within short inserts, as seen in Figure 7b. This revised search criterion sharply pushes sequences to the left and reveals that most (99.5%) sequences have less than 30 bp matches to the contaminant in their good region. We can improve the separation even further by looking at the *fraction* of the good region of each sequence involved in contaminant matches. The result, as shown in Figure 7c, is a clear separation of the only vector insert, which has a 47% match to the contaminant in its good region. This is about one in five thousand input sequences. The other sequences all have less than 10% match in their good regions. However, some short inserts may still be dropped because they have less than the `minimum` good sequence length. We will explain this further in the following Section **Discussion**. The `vector_cutoff` (20%) is used to separate contaminants from good sequences.

Readers should note that although this simple tag-matching method is adequate to recognize vector contaminants, a much more sophisticated algorithm is needed to screen successfully for larger contaminants such as *E.coli*, yeast, or human. Potentially, we can increase the tag size used and check the sequential ordering of matched tags to obtain more conclusive evidence of contaminant matching. This can be done similarly to the secondary sequence extension step explained previously. However, this larger contaminant screening capability has not been built into LUCY yet. In TIGR, we use another tool, the `dds.btab` program derived from the AAT package (Huang *et al.*, 1997), to screen for larger contaminants after LUCY processing.

[†] Among the 5022 sequences, 310 have less than `minimum` good sequence length and are not included in the statistics.

SUMMARY OF PARAMETERS AND OTHER SYSTEM CONSIDERATIONS

LUCY's control parameters are summarized in Figure 8. Most of these parameters have been discussed before, except the following:

- The three `pass_along` values define the minimum, maximum and medium clone lengths of a particular library. Usually, the preparer of the library knows these values. They are critical to the fragment assembly program in bridging contigs and constructing scaffolds, but LUCY does not interpret these values. It just passes them along with the output sequences.
- After all kinds of checking, comparing and trimming, the good region of a sequence must still be longer than the `minimum` good sequence length (100 bp) for it to be considered useful to the subsequent data processing stages. We do not want our assembly program to receive many small, trashy fragments.
- If users have multiple CPUs in their computers, they can dramatically increase LUCY's speed by telling it to run `xtra` execution threads concurrently. For example, on a Quad-CPU computer, LUCY's execution time can be cut to a quarter of its time spent on a single CPU computer. By default, LUCY runs just one thread.
- Users can tell LUCY to use a specific set of `output` file names, to be `quiet` during processing and only report errors, to `inform` them of names of sequences that are dropped due to low quality concerns or are resurrected by secondary sequence extension, and to produce a `debug` file listing the computed tag values (shown in Figure 2) for each sequence. These tag values reveal how LUCY arrives at a trimming decision for each sequence.

All parameters of LUCY are explained in greater detail in the document that comes with its distribution.

LUCY does not access (nor depend on) a database server. It is a design decision to separate LUCY from any site-specific infrastructure assumption. In TIGR, we use a separate program, RICKY, to drive LUCY and

Parameters	Default values	Related operation steps
pass_along min_value max_value med_value	0 0 0	pass to assembly program
error max_avg_error max_error_at_ends	0.025 0.02	quality area determination
window window_size max_avg_error ...	50 0.08 10 0.3	quality area determination
bracket window_size max_avg_error	10 0.02	quality area determination
range area1 area2 area3	40 60 100	vector splice site trimming
alignment area1 area2 area3	8 12 16	vector splice site trimming
vector vector_sequence_file splice_site_file	none	vector splice site trimming
cdna [min_span max_error initial_search_range]	none or 10 3 50	poly-A/T trimming
keep	none	poly-A/T trimming
size vector_tag_size	10	contaminant removal
threshold vector_cutoff	20	contaminant removal
minimum good_sequence_length	100	overall quality control
xtra cpu_threads	1	overall program control
output, quiet, inform_me, debug	none	overall program control

Fig. 8. Summary of LUCY parameters and their default values.

provide input/output bridging between LUCY and our database server. Since programs like RICKY are built around a specific infrastructure that varies from institution to institution, users need to design their own solutions to automate the process of running base-calling software, getting input to LUCY, converting LUCY's output to SQL update commands, and finally uploading LUCY's output to their databases. RICKY will not be useful outside TIGR's environment. LUCY, on the other hand, does not depend on any other program to run. Users can operate LUCY by manually providing its input data, as is often done in smaller research laboratories.

LUCY reads its input data from multi-FASTA text files, and writes its output also to multi-FASTA files. A multi-FASTA file is simply the concatenation of many FASTA sequences (National Center for Biotechnology Information, 2001), like those generated directly by *phred*. Each DNA sequence in a multi-FASTA file is delimited by a header line that begins with the greater-than symbol '>' followed by the sequence name, and may include other information about the sequence. LUCY records only the sequence name information which is terminated by the first space character that appears on the header line. The header line is followed by several more lines of data making up the actual DNA sequence, which can be in either upper or lower case letters. LUCY accepts the standard IUB/IUPAC DNA codes including the ambiguous letters (e.g. N for unknown bases). For each base in a DNA sequence, there must be a corresponding number denoting its quality value in the companion quality sequence. Quality sequences share the same header line format but are made up of numbers separated by spaces. Blank lines are not allowed in the input files and all lines must be shorter than 256 characters. Each input sequence is terminated by either the beginning of another header line or by the end of the input file.

LUCY makes no assumption about the order of sequences in the three input files: the first sequence file, the companion quality value file, and the optional second sequence file. As long as all necessary information can be found, DNA and quality sequences can be in different order in the first two input files. LUCY's output will always be in the order of the first sequence file, so a trivial 'feature' of LUCY is to sort quality sequences with the DNA sequences. A DNA sequence without its companion quality sequence or vice versa will be reported as an error. The second sequence file is allowed to have missing sequences that appear only in the first sequence file. Sequences that appear only in the second sequence file will be ignored by LUCY. The following is a typical output sequence from LUCY, where the header line contains this information: the sequence name, the three `pass_along` values of the minimum, maximum and medium clone lengths of the library, and the left and right trimming positions determined by LUCY.

```
>GCCAA03TF 1500 3000 2000 43 490
AGCCAAGTTTGCAGCCCTGCAGGTCGACTCTAGAGGATCCCCAGGATGATCAGCCACATT
GGGACTGAGACACGGCCCAAACTCCTACGGGAGGCAGCAGTGGGGAATCTTGCGCAATGG
GCGAAAGCCTGACGCAGCCATGCCGCGTGAATGATGAAGGTCTTAGGATTGTAAAATTCT
TTCACCGGGGACGATAATGACGGTACCCGGAGAAGAAGCCCCGGCTAACTTCGTGCCAGC
...
```

At TIGR, we run LUCY on a Sun workstation running the Solaris operating system (Version 5.5.1). LUCY has also been compiled and run successfully under the Linux operating system on a PC. Almost all ANSI C compilers should be able to compile LUCY source code since it is programmed using only standard C libraries and header files. LUCY has not been run on the MacOS or Windows by us, but we believe porting it to these two platforms should not be too difficult since all source code are included in its distribution. It is very likely that LUCY can run without any significant modification inside a Windows command shell.

LUCY is a CPU-bound program, not a memory-bound program. LUCY's static memory requirement grows very slowly with the number of input sequences, at roughly 60 bytes plus the sequence name storage for each input sequence. This is because LUCY does not store any sequence data in memory after processing them. To save memory, LUCY uses direct file addressing to access each sequence when it is needed, and stores only those pointers in the memory. Therefore, by all practical considerations LUCY can handle any number of input sequences. The dynamic memory requirement of LUCY is proportional to the longest sequence in the input data, *not* the number of input sequences, but its actual size varies from one processing step to the next.

The slowest step in LUCY is the vector trimming step which employs a quadratic time dynamic programming algorithm to find vector fragments in each sequence. To speed up LUCY's processing, multi-threading capability has been built into LUCY as introduced above. On a Linux PC with four CPUs running at 500 Mhz, it takes LUCY just 1 min and 12 s to process 5022 raw input sequences. When using one CPU, LUCY's processing time will be roughly four times that. This performance is considered acceptable to most applications. Also related to vector trimming, it is important that users do not turn on *phred*'s quality trimming function when doing base-calling. This often cuts the resulting sequences short and prevents LUCY from seeing the vector fragments. Leaving the trimming to LUCY will give it the information it needs to do its job well.

We cannot provide any generally applicable statistics of the trimming results of LUCY because these certainly depend a lot on input data quality, contaminant contents and LUCY's parameter settings. We can, however, provide some information specific to the 5022 test sequences we mentioned earlier that are obtained from TIGR's *Arabidopsis* genome project (Lin *et al.*, 1999). With default LUCY parameters, 310 of the input sequences are discarded because their good regions are shorter than the `minimum` length of 100 bp after trimming. Of the remaining 4712 useful sequences, their total length is 4148 326 bases, and their total good region length is 2072 533 bases. Therefore, about 50% of the raw data are trimmed away by LUCY. The resulting good regions have a base-call confidence level of above 97.5%, and contain no vector fragments or contaminants. Among the 5022 input sequences, only one is suspected to be a vector insert.

Although many major genome sequencing centers have already developed their own automatic data processing pipelines to perform some of the tasks LUCY is designed to do (Veklerov *et al.*, 1996; Smith *et al.*, 1997; Dear *et al.*, 1998; Wendl *et al.*, 1998), we believe LUCY can still be a useful addition to the freely available bioinformatic software for genomic data processing. LUCY's simple interface requirements allows it to be easily incorporated into any existing pipeline. LUCY is written in ANSI C code so it may improve certain processing steps when compared to Perl or script based solutions. Additionally, since LUCY is independent of any database infrastructure, it can be used automatically or manually, by both large and small research laboratories.

DISCUSSION OF SPECIAL CASES

We have given separate descriptions of LUCY's major processing steps. However, the actual processing within LUCY is far more complex than what has been explained. Often times it is the joint result from these processing steps that is used to make the final decision about the fate of a sequence. Trimming vector splice sites in the low quality region of sequences proves to be a very difficult problem. Since version 1.03 of LUCY (now at 1.16), all new improvements were made possible because people using LUCY discovered abnormal trimming results with their eyes and provided them to us for inspection. We found that solutions for many of these special cases can contradict each other, meaning that making LUCY handle one special case may introduce new problems in sequences that had been handled correctly before. We present some examples in this Section **Discussion** to illustrate the difficulty of the problem. This presentation is not meant to include all special cases that can happen during LUCY processing.

To start with, LUCY always looks at the initial search areas for any vector fragment (Figure 9a). Under normal conditions, there are some low quality bases (20–50 bp) at the beginning of a sequence. Vector fragments are well within the initial search areas and can be identified even within the low quality region. The rest of the sequence is then searched against the downstream splice site sequence to guard against short inserts. Usually the clone is much longer (2 kbp) than the sequencing machine can read at once (<900 bp), so LUCY will not find any vector fragment at the other end of a sequence read. Most input sequences (well over 90%) fall into this case.

Sometimes, a low quality region can separate two good regions, as seen in Figure 9b. This may be the result of incidents such as power interruption caused by lightning during sequencing runs[†]. In this case, LUCY finds the longest high quality region and uses that as the good region. Although the other good regions, as long as they are longer than `minimum` good sequence length, could still be used separately, this would require LUCY to generate new sequence names that could cause some database tracking nightmares (e.g. what subsequences come from which original sequence, etc). The extremely low frequency of

[†] This is no joke; it actually happened a few times in TIGR during thunderstorm seasons.

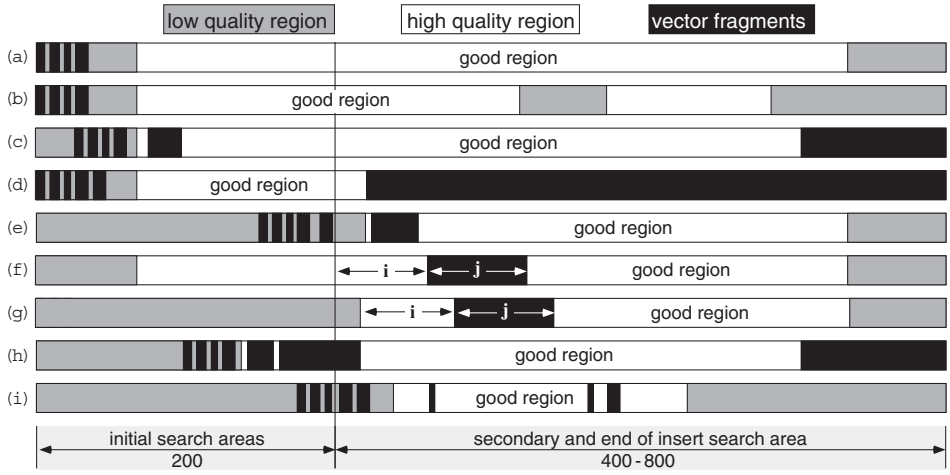

Fig. 9. Examples of special cases of sequence cleaning that LUCY has been designed to handle. Although we plot each example using the same length sequence for esthetic reasons, it should be understood that each sequence can be of variable length and can contain more than one problematic scenario. See the main text for explanation of each case.

this occurrence, and the nature of the shotgun fragment assembly principle, make the effort to salvage the shorter good regions in a sequence unnecessary.

Vector fragments may not always be in the low quality region. Sometimes, they run into the good quality region as well. The other end of the vector splice site may also be found towards the end of a sequence, as shown in Figure 9c. If the sequence is a short insert, the length of the good region may be much shorter than the length of the sequence, as exemplified in Figure 9d. Normally, LUCY will drop a sequence whose good region is shorter than `minimum` good sequence length to prevent useless data from clogging the subsequent data processing stages.

Sometimes, the low quality region runs longer than the initial search areas. In this case, as shown in Figure 9e, LUCY has to make a second attempt to search for vector fragments beyond the initial search areas, in order to find any additional vector fragments inside the high quality region. The reason for doing this is because some of the low quality bases may mask the true identity of vector fragments. In order to guarantee that no vector fragment is included in the reported good region, LUCY must search part of the high quality region beyond the initial search areas as well. This is the second *adaptive* vector splice site search we mentioned previously.

If LUCY does not find any vector fragment within the initial search areas, it may be the case that the sequencing PCR reaction started well before the splice site. To guard against this, LUCY needs to conduct a second search attempt beyond the initial search areas as well. However, this may create another problem, as shown in Figure 9f. What if LUCY finds a short match to the splice site sequence inside the high quality region, but no other vector fragments are found near it to confirm the match? Should LUCY treat it as a positive identification of a vector fragment and trim the sequence accordingly, or should LUCY consider it a match by random chance that can be ignored? There is no absolute correct answer to this. Therefore, LUCY employs a heuristic judgment: if the distance between the vector match and the end of the search areas is shorter than the width of the vector match itself, i.e. $i < j$ in Figure 9f, then LUCY declares it a vector fragment and trims it. Otherwise, LUCY rejects it as a random match. Additionally, if the vector match is equal or more than half the total length of the good region ($j > 1/2$ length of good region), then LUCY will trim it as well. Since the search areas are 200 bp long, it basically means that LUCY will trim all singular vector matches longer than the minimum `alignment` length in each area, but will ignore most short vector fragment matches in the middle of a long high quality region.

The heuristic above means that even if LUCY does throw out some false-positive vector fragments, they will all be limited to the first 200 bases of a sequence, and LUCY needs much more concrete evidence before it will accept a vector fragment match inside the high quality region. In addition to these, the heuristic has to be augmented for additional special cases where the low quality region extends much into the sequence and goes beyond the initial search areas, as shown in Figure 9g. In this new case, the boundary of heuristic judgement will be shifted to the right end of the low quality region. The reason for this is similar to case 9e, to prevent any vector fragment from getting into the good region selected by LUCY.

Another special case is when LUCY finds a vector fragment that extends right up to the end of the initial search areas, as seen in Figure 9h. Here, the end of the vector fragment may be the actual boundary of it, or it may simply be an artificially created boundary because of the fact that LUCY searches for vectors in the first 200 bases first. To confirm the actual ending of the vector fragment, LUCY has to conduct the second search as well[†]. Finally, even when splice site trimming is done normally and there is a good region inside a sequence, if the good region length is less than the `minimum` good sequence length after the length of matched vector fragments are subtracted from it, LUCY will still reject the sequence in fear of introducing contaminants into the data processing pipeline. Matched vector fragments, as shown in Figure 9i, are found during the quick contaminant removal step.

Although these special cases are also discussed separately, it is not necessary that they occur separately. More often for low quality sequences, they actually occur simultaneously. For example, a sequence can have a short insert, a long initial low quality region, another low quality region separating two good quality regions, and a single vector fragment match inside its good region with no other vector fragment found in the initial search areas due to numerous base-calling errors. LUCY has been made to handle most, if not all, possible combinations of special situations together.

CONCLUSION

Although there are many DNA sequence comparison and analysis algorithms in the bioinformatics literature, our experience has been that to make them function correctly inside a program that processes real-world data, the programmers must make an extra effort to consider special cases reported by users. This is especially true when the amount of data is large and the quality of data can be low, as in the sequence cleaning stage. We believe that this is not just an isolated situation for LUCY. It is, indeed, a common phenomenon that arises during many bioinformatic software development cycles. Due to the inherently unpredictable nature of biological data, there is always some distance between the theorectical design of a bioinformatic solution and the successful implementation of the solution in a working program that can handle real-world data reliably. The only way to shorten such distance to perfection, in our opinion, is to form a close collaboration between computer scientists and biologists. This allows the wisdom and experience of biologists to be slowly translated into functional program code.

Further improvements to LUCY may prove difficult, as LUCY now strikes a delicate balance among the various strategies for dealing with real-world problems that have been encountered at TIGR. If LUCY is able to do a better job today than its previous versions, it is all because we have a few hardworking biologists who actually talk to computer scientists.

ACKNOWLEDGEMENTS

We wish to thank Dr Granger Sutton and Dr Tony Kerlavage for leading us into this difficult but intriguing problem. We would also like to thank Ms Anna Glodek, Mr John Scott and Mr Terrance Shea for providing us their valuable suggestions when we were developing LUCY. Finally, we thank Dr Steven Salzberg and Dr Claire Frasier for generously allowing us to make this program freely available to the academic community.

REFERENCES

Dear,S., Durbin,R., Hillier,L., Marth,G., Thierry-Mieg,J. and Mott,R. (1998) Sequence assembly with CAFTOOLS. *Genome Res.*, **8**, 260–267.

Ewing,B. and Green,P. (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.*, **8**, 186–194.

Ewing,B., Hillier,L., Wendl,M.C. and Green,P. (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res.*, **8**, 175–185.

Huang,X., Adams,M., Zhou,H. and Kerlavage,A. (1997) A tool for analyzing and annotating genomic sequences. *Genomics*, **46**, 37–45.

Lin,X. *et al.* (1999) Sequence and analysis of chromosome 2 of the plant *Arabidopsis thaliana*. *Nature*, **402**, 761–768.

National Center for Biotechnology Information (2001) FASTA file format specification. Available on the Internet. http://www.ncbi.nlm.nih.gov/blast/html/search.html

Paracel (2000) TRACETUNER, capturing the most information from the latest DNA sequencing systems. Information on the Internet. http://www.paracel.com/html/tracetuner.html

Smith,T.M., Abajian,C. and Hood,L. (1997) HOPPER: software for automating data tracking and flow in DNA sequencing. *Comput. Appl. Biosci. (CABIO)*, **13**, 175–182.

Veklerov,E., Eeckman,F. and Martin,C. (1996) MTT: a software tool for quality control in sequence assembly. *Microb. Comp. Genomics*, **1**.

Wendl,M., Dear,S., Hodgson,D. and Hillier,L. (1998) Automated sequence preprocessing in a large-scale sequencing environment. *Genome Res.*, **8**, 975–984.

[†] This used to be a bug in previous versions of LUCY.