Supplement to: Sequence-based heuristics for faster annotation of non-coding RNA families

Zasha Weinberg* and Walter L. Ruzzo†

October 12, 2005

1 Introduction

This paper provides supplementary technical details and results, specifically:

- How to set heuristic thresholds.
- When can heuristic filters be better than rigorous ones.
- How to calculate ROC-like curves for BLAST.
- How to calculate ROC-like curve for profile HMMs for very large databases like RFAMSEQ.
- Why it is important to train from the CM and not the MSA.
- How to extend the ML-heuristic technique to fully general CMs, as opposed to the simplified model considered in the main text.
- A simple experiment evaluating the ML-heuristic as an improved SECIS element predictor.
- Complete set of ROC-like curves for all 8 families tested.

2 How to set heuristic thresholds

The paper briefly described an approach to select a heuristic threshold: find a threshold yielding a small filtering fraction like 0.01. In general, when the filtering fraction gets very small, the profile HMM scan time will start to dominate the overall scan time; therefore there is little incentive to reduce the filtering fraction further. On the other hand, high filtering fractions require that the CM is run too often, making scan time prohibitive. Therefore, a roughly fixed filtering fraction is a rational choice. (There are many other reasonable strategies, e.g., finding a filtering fraction that allows the HMM to find all the seed members.)

*Dept. of Computer Science & Engineering, University of Washington, Seattle, WA, 98195, USA, zasha@cs.washington.edu

†Depts. of Computer Science & Engineering and Genome Sciences, University of Washington, Seattle, WA, 98195, USA, ruzzo@cs.washington.edu

To calculate a heuristic threshold yielding a desired filtering fraction, we can use the same technique used for generating ROC curves. The simplest approach would be to run the profile HMM on the entire database to collect inclusion points, then find the appropriate heuristic threshold, and re-scan. Even if the profile HMM scan takes half of the overall scan time, this would only increase total scan time by 50%. However, it does lead to very large files when run on large genome databases given the current techniques for calculating ROC curves (see below for details).

An alternative is to scan some representative sequences, and estimate based on that. We have observed that database G+C content tends to account for much of the variation in filtering fraction [14, supplement]. Therefore, we use three test genomes, one with very low G+C content, one with very high G+C, and one in the middle. We assume that the database contains equal portions of these three genomes. (Other assumptions about composition are also possible, and more test genomes can be used.) Each genome is scanned, storing inclusion points. Then the computer iterates through heuristic thresholds starting at the highest (most stringent). At each proposed heuristic threshold, the inclusion points allow a quick calculation of the filtering fraction for each of the three test genomes. Based on the assumption about the composition of the database, it is easy to estimate the filtering fraction on the overall database, extrapolating from the test genomes. The iteration over heuristic thresholds stops when the threshold is found that is estimated to yield the desired filtering fraction on the genome database. (A binary search strategy would reduce the time taken by this step. However, the process of collecting inclusion points is linear in the number of nucleotides scanned, so a binary search strategy cannot reduce overall CPU time significantly.)

3 When can heuristic filters be better than rigorous ones

This section demonstrates an example where a rigorous filter can fail to discriminate well compared to a heuristic filter. Although the example is highly simplified, it demonstrates the basic notion: since rigorous filters must optimize for rare RNAs (in order to guarantee perfect sensitivity), they may not discriminate as well as heuristics, which optimize for the common case. (Note: poor discrimination does not negate rigorousness. Consider a filter that never eliminates anything, but rather runs the CM on the full sequence database. Such a filter is rigorous, because it never eliminates anything, but clearly it does not discriminate well between positives and negatives.)

Our input MSA in the example consists of three sequences: GC, GU and AU, where the two nucleotides form a base pair. (Obviously this is not biologically plausible; it is just intended to demonstrate the principle.) We will discuss the example fully in terms of probabilities (rather than the logarithmic scores used in our earlier papers on rigorous filters). For simplicity, we also ignore pseudocounts.

The CM for this scenario is as follows: $\mathrm{S}_1 \rightarrow g\mathrm{S}_2c|g\mathrm{S}_2u|a\mathrm{S}_2u; \quad \mathrm{S}_2 \rightarrow \epsilon$. All rules for S_1 have probability 1/3. We assume for the following that the CM's probability threshold is set at 1/3, since this would enable it to find all three sequences.

We now consider HMM probabilities for a rigorous filter (using the infinite-length forward algorithm [13]) and for a heuristic filter (using the ML-heuristic). For convenience, let g be the probability of G occurring in the first position, a be the probability of A in the first position, c be the probability of C in the second position and u the probability of U in that position. (Note that in our example each nucleotide occurs in exactly one of the positions.)

ML-heuristic. With the ML-heuristic, $g = 2/3$ and $a = 1/3$, since those are the frequencies in the first position. Similarly, $u = 2/3$ and $c = 1/3$. Therefore, according to this HMM, the probability of the sequence GC is $\Pr(\text{GC}) = 2/9$, while $\Pr(\text{GU}) = 4/9$ and $\Pr(\text{AU}) = 2/9$. Due to the limitations of profile HMMs, the sequence AC will have non-zero probability. With the ML-heuristic, $\Pr(\text{AC}) = 1/9$. Fortunately, with heuristic probability threshold 2/9, only the training sequences GC, GU and AU will be passed to the CM, while the anomalous AC will be eliminated by the filter.

However, note that this profile HMM is not rigorous. For example, if we use the CM's threshold of 1/3, then GC and AU pairs are missed. Heuristic filters make no guarantees about rigorousness, but with the appropriate threshold (2/9), discrimination is excellent.

Rigorous filter. For the rigorous filter, we obtain three inequalities: $gc \geq 1/3$, $gu \geq 1/3$ and $au \geq 1/3$. (Note that these are not linear inequalities, as in our previous paper, simply because we are working in probability space.) If A, C, G and U are expected in equal frequency, the infinite-length forward algorithm objective function is equivalent to $1/16(gc + gu + au + ac)$. The minimum is at equality, where $\Pr(\text{GC}) = \Pr(\text{GU}) = \Pr(\text{AU}) = 1/3$ (for which one solution is $a = c = g = u = 1/\sqrt{3}$). Unfortunately, using the CM's threshold of 1/3 (as a rigorous filter must), now $\Pr(\text{AC}) = 1/3$. Thus, the rigorous filter will pass AC to the CM.

In this example, the rigorous filter requires a higher filtering fraction than the ML-heuristic: it must accept AC in order to achieve 100% sensitivity, while the ML-heuristic can achieve this sensitivity without accepting AC. Moreover, the rigorous filter will have this problem with any threshold, and so its ROC curve will be worse than that of the ML-heuristic.

4 How to calculate ROC-like curves for BLAST

In computing ROC-like curves for BLAST, we assumed that any hit with E-value e will be found if and only if the E-value cutoff is set at e' for any $e' > e$. In fact, this is not strictly true.

BLAST uses exact word matches to seed searches, but also uses ungapped alignments to seed the more expensive gapped alignments. These ungapped alignments are treated at the same E-value cutoff, which may violate our assumption. For example, suppose we run BLAST at E-value cutoff 1000, and find a hit with E-value 5. If we run BLAST with E-value cutoff 10, we expect to again obtain this hit. However, the hit's E-value in the ungapped alignment phase may be greater than 10, and therefore the hit may not be discovered.

Clearly, this is unlikely to be a significant factor. In fact, it may even overstate BLAST's capabilities, since it effectively improves the sensitivity of BLAST's ungapped alignment phase.

To be sure, we ran BLAST with lower E-value cutoffs and found that our predicted sensitivity and filtering fractions differed from the actual values by an insignificant amount. For example, for RF00005 the graphs shown in the paper and this supplement were generated with an E-value cutoff of 1000. We re-ran RF00005 with a cutoff E-value of 1. The ROC-like curves for these two invocations of BLAST overlapped completely to the available resolution. Thus, empirically, the error introduced by the incorrect assumption about BLAST's E-values is not significant.

(A)

	A	C
A	1	0
C	0	0

(B)

	A	C
A	$\frac{2}{5}$	$\frac{1}{5}$
C	$\frac{1}{5}$	$\frac{1}{5}$

(C)

A	$\frac{2}{3}$
C	$\frac{1}{3}$

(D)

A	$\frac{3}{5}$
C	$\frac{2}{5}$

Table 1: Training heuristic HMMs: from CM or from MSA. See text for explanation. (A) straight counts of the sequences—one A-A pair. (B) CM probabilities with plus-one pseudocounts. (C) Probabilities of the left HMM position, using the ignore-SS heuristic and with plus-one pseudocounts (trained from the MSA). (D) Probabilities of the left HMM position, using the ML-heuristic (trained from the CM).

5 Calculating ROC-like curves with very large databases

The curves in this paper were generated by scanning the roughly 8 Gbase RFAMSEQ database from the Rfam database version 5.0. This yields about $1.6 \cdot 10^{10}$ inclusion points (since each nucleotide is scanned twice, once for each DNA strand). To reduce computer memory demands, we stored these in an associative array mapping an inclusion point (as a C/C++ 'float' data type) to an integer representing the number of times that inclusion point was observed. The associative array implementation used the C++ Standard Template Library 'map' class, which is based on red-black trees. Using this simple scheme, the memory requirements were still large—in some cases, we observed the process using over 1 GB of RAM—but were practical for these tests.

If necessary, the memory requirements could be reduced further by storing the inclusion points with less precision. This precision is probably not necessary for either accurate ROC-like curves, or for reasonable estimates of score thresholds for a given filtering fraction. Other data structures may also reduce memory requirements.

6 Training profile HMMs from the CM versus from the MSA

As described in the paper, a simple method to transform an MSA into an HMM is to erase the structure annotation, and run the standard algorithm to learn a profile HMM [3]. Unfortunately, when pseudocounts (or other priors) are used, the resulting profile HMM does not capture the same primary sequence information as the CM does. This phenomenon arises since the pseudocounts will affect the profile HMM differently from the CM. (This explains the poor performance of the ignore SS heuristic for families that do not contain enough sequences to overwhelm the pseudocounts.)

Example. We illustrate the issue with an example. Suppose we have an MSA with two columns, and these columns are base paired. For extra simplicity, we ignore insertions and deletions, and consider only the two nucleotides A and C (so we are assuming a two-letter alphabet). The MSA contains one sequence: an A-A pair. Finally, suppose we apply the simple plus-one pseudocount, i.e., Laplace's rule [3].

Table 1 summarizes the various calculations. In the CM, with plus-one pseudocounts, A-A will have probability $\frac{2}{5}$, while other base pairs will have probability $\frac{1}{5}$.

Now we apply the ignore-SS heuristic: we create profile HMM from the MSA, using the same plus-one pseudocount. In the first column (the left nucleotide of the base pair), the probability of A is $\frac{2}{3}$, and the probability of C is $\frac{1}{3}$.

Applying the ML-heuristic to the CM, the probability of A is $\frac{3}{5}$, and the probability of C is $\frac{2}{5}$. These probabilities are clearly different from those calculated under ignore-SS. The reason is that, since there are more events in the CM, the effective weight of the pseudocount is higher for CMs than for HMMs. The difference between $\frac{2}{3}$ (≈ 0.67) and $\frac{3}{5}$ ($= 0.6$) is modest, but can significantly affect accuracy. Moreover, if we had used all four nucleotides, with the same single sequence MSA of A-A, the probabilities would be $\frac{2}{5}$ ($= 0.4$) and $\frac{5}{17}$ (≈ 0.29).

Thus, when pseudocounts are used, ignore-SS yields different probabilities from the ML-heuristic. Although this example demonstrates that the two heuristics yield different results, the example does not indicate which is better.

Which is better? We conclude that the ML-heuristic is better, based primarily on empirical results. Empirically, our ROC-like curves show that the ML-heuristic is better than ignore-SS. Moreover, the cases in which their accuracy is comparable occur when there are many known family members, so the pseudocounts are irrelevant, and the two heuristics result in approximately the same probabilities.

From a theoretical perspective, there are reasons to believe that it is best for the profile HMM to follow the CM as closely as possible, and reasons to believe it should be independent (i.e., use priors that are ignorant of the CM's use of priors). We discuss each of these positions, and then analyze them.

For the argument in favor of independence, consider an extreme case: an oracle filter that perfectly eliminates everything but the ncRNA homologs. Such a filter is the ideal case, and will yield perfect sensitivity and specificity, even though it completely ignores the CM. Although this ideal filter is not attainable, we may claim that the ignore-SS heuristic, using profile HMM-appropriate priors, is better at finding ncRNAs, and therefore closer to the ideal filter. This argument, of course, requires that the profile HMM's priors are truly better for finding ncRNA homologs.

However, there are reasons (in addition to empirical results) to expect that following the CM is better (i.e., that the ML-heuristic should prove superior). First, ncRNAs detected by the profile HMM that are not detected by the CM are essentially useless; especially since we are evaluating sensitivity relative to the CM, it is advantageous to bias the profile HMM to the CM. Although, in the extreme case that the profile HMM is perfect, the perfect filter is ideal, this ideal filter is unlikely to be achievable in practice.

Second, the CM can encode information about ncRNAs that is not accessible to a standard profile HMM. This will be especially relevant for CMs using sophisticated, RNA-aware priors. For example, such a CM would reflect different mutation rates for nucleotides that are base paired versus unpaired nucleotides; in some cases, this distinction may be encoded in a prior. If a profile HMM ignores the priors, it will be losing this information. The ML-heuristic, by basing its probabilities directly on the CM, exploits any priors used in the CM. Better priors trained on RNA alignments are already used in the RSEARCH program [5], and we anticipate further developments in this area.

In summary, there are theoretical arguments to support either ignore-SS or ML-heuristic. We find the ML-heuristics more compelling because it is unclear whether the ideal filter proposed in the ignore-SS argument will be relevant in practice. Moreover, the ML-heuristic argument will be more significant as CM priors are improved. Overall, the empirical evidence supporting the ML-heuristic is the most persuasive evidence.

7 ML-heuristic with fully general CMs

7.1 Handling general CM grammars

The ML-heuristic uses precisely the same profile HMM grammar as the rigorous profile HMM filters designed in previous work [13]. In those terms, the ML-heuristic uses the *compact-type* HMM. (The alternative is the *expanded-type* HMM, which uses additional states. These additional states slow scan speed slightly, but for rigorous filter often reduce the filtering fraction very significantly. However, for the ML-heuristic, expanded-type HMMs appeared to offer no accuracy benefit, as demonstrated in figures 3 and 4.)

The computation of the HMM probabilities for the ML-heuristic is somewhat more complicated in the case of fully general CMs. With general CMs, the profile HMM grammar includes insert states, which have self loops. Also, the $\overline{\mathrm{S}}_i^L$ and $\overline{\mathrm{S}}_i^R$ states are split into separate states based on whether they are emitting (emitting a, c, g or u) or a non-emitting deletion state (emitting ϵ). The CM states S_i are split into 2 or as many as 4 states in a similar manner.

A consequence of these additions is that a given CM state need not be entered when it generates a sequence in the infinite MSA. Also, some CM states—namely the insert states—may be entered more than once when generating a sequence. To account for this, we calculate the expected number of times that each CM state will be entered in generating one sequence.

Another consequence is that CM parses become more complicated. General CMs are conceptually divided into *nodes*. A node relates to the S_i in this paper, but each node has several states, including one or two insert states. We therefore consider *sub-parses*, which start at one CM node and end at the next. Each sub-parse consists of one or more CM rules. The CM rules can be mapped to the corresponding profile HMM rules, i.e., the profile HMM rules that would be used if the infinite MSA is parsed without structure.

We now define the method to set profile HMM probabilities for the ML-heuristic in the general case. Let $\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}}$ be any profile HMM rule. Let $\pi(\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}})$ be the set of CM sub-parses that would use profile HMM rule $\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}}$. For any one sub-parse π, let π_1 be the first state used in the parse. Let $E(\pi_1)$ be the expected number of times this first state is entered, and $\Pr(\pi)$ be the probability of the parse assuming this first state is entered, i.e., simply the product of the probabilities of the rules in π.

Then, the count for profile HMM rule $\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}}$ is:

$$\mathcal{C}(\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}}) = \sum_{\pi \in \pi(\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}})} E(\pi_1) Pr(\pi)$$

This equation yields the expected number of times the profile HMM rule will be used to parse a sequence from the infinite MSA without structure. We note that the set $\pi(\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S'}})$ is infinite in size because it can include an unbounded number of insert state self loops. Fortunately, the contribution of the unbounded series of insert states is easily summed, since it is simply the sum of an infinite geometric series.

The equation does not, however, set profile HMM insert state self loop rules correctly, since profile HMM insert states can be used multiple times within one sequence. So, for insert states, the following rule is used:

$$\mathcal{C}(\overline{\mathrm{S}} \rightarrow x\overline{\mathrm{S}}) = E(\mathrm{S}) \Pr(\mathrm{S} \rightarrow x\mathrm{S})$$

where S is the CM state corresponding to HMM insert state $\overline{\mathrm{S}}$.

Once these virtual counts $\mathcal{C}$ are computed, they are normalized into probabilities as before.

7.2 Odds ratios vs. probabilities

We have described the ML-heuristic under the assumption that CMs are straight probabilistic models. However, in fact their emission scores are set from odds ratios. Typically, the numerator in the odds ratio is the CM-assigned probability, and the denominator (null model) is a 0th-order Markov model. Our technique generalizes to this case easily: we normalize the CM probabilities to extract the straight-CM probabilities (the numerator), then create an ML-heuristic profile HMM. Finally, we apply the null model (the denominator) to the profile HMM, to convert it to odds ratios.

8 ML-heuristic as an improved SECIS element predictor

The paper's results showed that the ML-heuristic is 9 times more sensitive than BLAST in finding SECIS elements (Rfam ID RF00031), which suggested that filtered CM solutions may help to build a better SECIS element finder.

The SElenoCysteine Insertion Sequence (SECIS) element directs the incorporation of the "21st" amino acid selenocysteine. In eukaryotes, when an mRNA includes a SECIS element in its 3' UTR, UGA codons code for selenocysteine residues, instead of their usual STOP function. Proteins with selenocysteine residues are called *selenoproteins*.

Selenoprotein annotation is important for two reasons. First, due to the unusual use of the UGA codon, current protein-coding gene prediction programs cannot hope to correctly predict selenoproteins' exons.

Second, selenoproteins have medical significance in that they are the major context in which cells use selenium [12, 6]. There is strong evidence that selenium deficiency causes male infertility specifically due to compromised function of a selenoprotein, PHGPx, that functions in sperm development [9]. Less direct evidence suggests possible roles for selenoproteins in, for example, cancer [4, 11] and brain function [10].

Recent work screened various organisms for selenoproteins, seeding searches based either on a predicted SECIS or on conserved ORFs with internal UGA codons, and conservation continuing after the UGA [7, 1, 8]. Improved SECIS element prediction may help to discover new selenoproteins, and to relax constraints elsewhere in the pipeline.

Previous work on detecting selenoproteins has used the program SECISearch [7]. Basically, this program uses the PatScan program [2] to detect a SECIS-like pattern, then folds candidates using Mfold [15] using an energy threshold and imposing other constraints on the predicted structure to reduce the false positive rate.

We compared CMs with ML-heuristic filters on Rfam family RF00031 to SECISearch 2.1. To assess sensitivity, we used 61 GenBank entries containing known selenoproteins in the SelWorld Database (`http://www25.brinkster.com/selworld/index.asp`). Many entries in this database were undoubtedly used directly or indirectly in both SECISearch and in the Rfam SECIS family, but some sequences are new. To assess selectivity, we ran both programs on random sequences generated with uniform, independent probabilities of A, C, G and T. For robustness, we estimated false positive rates based on at least 10 predictions in random data. (The sequences tested were 5 Kbases. We tested sequences until SECISearch had predicted SECIS elements in 10 sequences. We ignored multiple predictions in the same sequences, since typically these are redundant hits with slightly different nucleotide positions.)

Under default settings, SECISearch found SECIS elements in 30/61 SelWorld sequences. It predicted 1 false positive per 0.6 Mbases. Filtered CMs found SECIS elements in 34/61 SelWorld

entries, and predicted 1 false positive per 16.7 Mbases. We also tested a "looser" SECIS pattern supplied with SECISearch 2.1. With this setting, SECISearch found 35/61 SECIS elements in SelWorld, but its false positive rate was much worse—1 per 0.036 Mbases— than with its default pattern.

We conclude that at comparable levels of sensitivity, the CM solution is over 20 times more selective than SECISearch 2.1 in its default settings. This preliminary experiment is somewhat crude, e.g., variation in C+G content is not considered, nor is a more realistic random model of genome sequences. However, it is difficult to explain a factor of 20 difference by such issues.

It would be promising to integrate a CM-based SECIS element finder into a selenoprotein detection pipeline. It is possible that even better ML-heuristic/CM performance could be realized by improving upon Rfam's SECIS element MSA (i.e., that of family RF00031).

9 Complete set of ROC-like curves

There are 16 figures in this supplement, two for each family discussed in the paper. One figure uses log scale for plotting 1-sensitivity, to show extra detail at high levels of sensitivity. The other does not. Both of these graph types were shown in the paper.

The curve labels are as follows. "ML-heur" or "ML path" is the ML-heuristic profile HMM. "Rigorous HMM" is the expanded-type rigorous profile HMM. "ignore-SS" is the ignore-SS profile HMM.

For BLAST, the set of known ncRNAs used is either "SEED" or "90%". "SEED" means the list of known ncRNAs was used on which the CM was trained—this is the normal case. "90%" means that the full set of ncRNAs found with BLAST+CM on the Rfam site was used, but sequences were discarded that were more than 90% similar. (This corresponds to the sequences in Rfam.fasta.gz. This gives BLAST an unfair advantage, but it's interesting to see the results.) "1-sided" means that whenever BLAST found a hit, only the hit's DNA strand would be scanned by the CM. (This is the scheme used in the main paper.) With "2-sided", both strands of a BLAST hit are scanned. (This is the current technique in the Rfam database.)

"ML-heur (full 90% id)" is the ML-heuristic trained on the 90% of the full ncRNA family—the same set as for "90%" in the BLAST curves.

Some scans were not done; in this case, the curves are missing from the graphs.

The graphs support a number of conclusions:

- The expanded-type ML-heuristic does not discriminate better than the compact-type. We compared them for Rfam's tRNA family (RF00005; figs. 3 and 4, "ML path" and "expanded ML path"). Since the lines appear to overlap, there was no significant difference in performance on ROC-like curves for the two profile HMM types.

- The ignore-SS heuristic is inferior to the other profile HMMs when few seed members are available. For example for RF00010 (figures 5 and 6), ignore-SS requires filtering fractions that are about 10^4 times higher than those of the other techniques for the same level of sensitivity. When more training examples are available, ignore-SS produces comparable results, e.g. RF00005 (figures 3 and 4).

- The ML heuristic outperforms rigorous profile HMMs in most cases. In some cases, the two heuristics' curves approximately overlap. For example, for RF00001 (figure 1) they

overlap almost entirely, although figure 2 shows that at the highest levels of sensitivity, the ML-heuristic is better. In many families, the ML-heuristic is generally better, particularly RF00031 (figures 9 and 10; note that the rigorous filter is dark blue in these figures).

For the tRNA family, their relative performance varies: for most of the ROC-curve the ML-heuristic outperforms rigorous filters (figures 3 and 4), but at very high sensitivity and filtering fractions close to 1, the rigorous filters are superior (figure 4). Overall, the ML-heuristic appears better in terms of sensitivity measured by ROC-like curves.

- 1-sided BLAST searching is better in discriminative power than 2-sided BLAST. In every ROC-like curve, the 1-sided heuristic is superior at every point to the 2-sided heuristic, often by a factor close to 2.

  This is intuitive, since when BLAST detects an approximate match on one strand, there is no biological reason to necessarily expect a hit on the opposite strand, unless BLAST also finds an approximate match on that strand. Therefore, up to half of the sequence returned by a 2-sided BLAST filter has little evidence for an ncRNA hit. We note that it is easier to implement a 2-sided BLAST heuristic, because a sequence can directly be submitted to a CM scanning program.

- With both profile HMMs and BLAST, we used either the seed members or the "full 90% id" set. The use of the "full 90% id" set yields an unrealistic advantage, since in practice only the seed members would be known.

  We tested the "full 90% id" for profile HMMs only for Rfam's tRNA family (figures 3 and 4). At the relatively high levels of sensitivity that are of interest, the "full 90% id" profile HMM does better than the profile HMM trained only on the seed members.

  BLAST took more advantage of the "full 90% id" set, in that the difference between its performance with either set was larger. A particular difference was for Rfam's tRNA family (figures 3 and 4), where BLAST was able to take dramatic advantage of the "full 90% id" set, particularly at high levels of sensitivity.

References

[1] Sergi Castellano, Sergey V. Novoselov, Gregory V. Kryukov, Alain Lescure, Enrique Blanco, Alain Krol, Vadim N. Gladyshev, and Roderic Guigó. Reconsidering the evolution of eukaryotic selenoproteins: a novel nonmammalian family with scattered phylogenetic distribution. *EMBO Rep.*, 5(1):71–77, 2004.

[2] M. Dsouza, N. Larsen, and R. Overbeek. Searching for patterns in genomic data. *Trends in Genetics*, 13(12):497–498, 1997.

[3] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis.* Cambridge University Press, Cambridge, UK, 1998.

[4] Howard E. Ganther. Selenium metabolism, selenoproteins and mechanisms of cancer prevention: complexities with thioredoxin reductase. *Carcinogenesis*, 20(9):1657–1666, 1999.

[5] Robbie J. Klein and Sean R. Eddy. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4(1):44, 2003.

[6] Alain Krol. Evolutionarily different RNA motifs and RNA-protein complexes to achieve selenoprotein synthesis. *Biochimie*, 84(8):765–774, 2002.

[7] Gregory V. Kryukov, Sergi Castellano, Sergey V. Novoselov, Alexey V. Lobanov, Omid Zehtab, Roderic Guigó, and Vadim N. Gladyshev. Characterization of mammalian selenoproteomes. *Science*, 300(5624):1439–1443, 2003.

[8] Gregory V. Kryukov and Vadim N. Gladyshev. The prokaryotic selenoproteome. *EMBO Rep.*, 5(5):538–543, 2004.

[9] Matilde Maiorino, Valentina Boselloa, Fulvio Ursinia, Carlo Forestab, Andrea Garollab, Margherita Scapina, Helena Sztajerc, and Leopold Flohéc. Genetic variations of gpx-4 and male infertility in humans. *Biol Reprod*, 68(4):1134–1141, 2003.

[10] Ulrich Schweizer, Lutz Schomburg, and Nicolai E. Savaskan. The neurobiology of selenium: Lessons from transgenic mice. *J Nutr*, 134:707–710, 2004.

[11] Philip R. Taylor, Howard L. Parnes, and Scott M. Lippman. Science peels the onion of selenium effects on prostate carcinogenesis. *J Natl Cancer Inst*, 96(9):645–647, 2004.

[12] C. D. Thomson. Assessment of requirements for selenium and adequacy of selenium status: a review. *Eu J Clin Nutr*, 58(3):391–402, 2004.

[13] Zasha Weinberg and Walter L. Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. In *Proc. Eighth Annual Inter. Conf. on Computational Molecular Biology (RECOMB)*, pages 243–251. ACM Press, 2004.

[14] Zasha Weinberg and Walter L. Ruzzo. Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20(suppl. 1):i334–i340, 2004.

[15] M. Zuker. Computer prediction of RNA structure. *Methods in Enzymology*, 180:262–288, 1989.

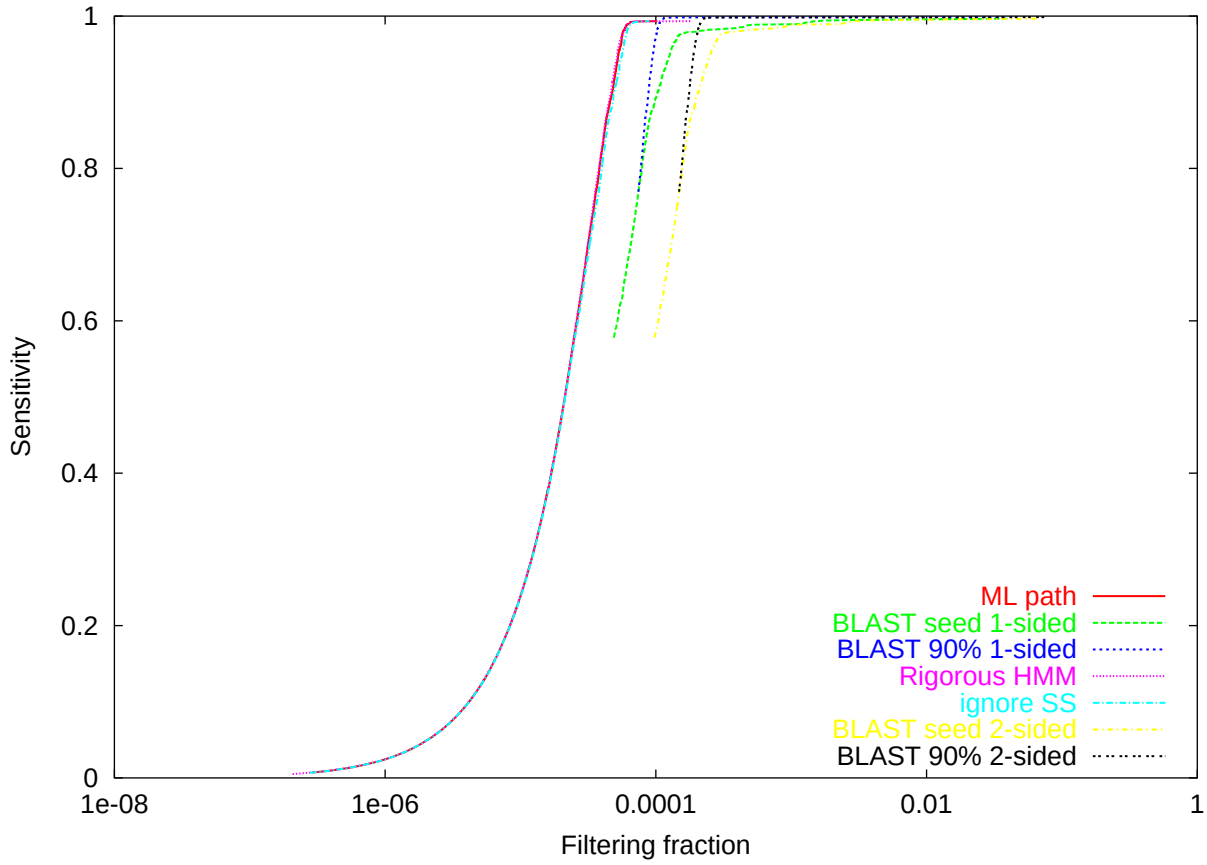

Figure 1: ROC-like curve: RF00001

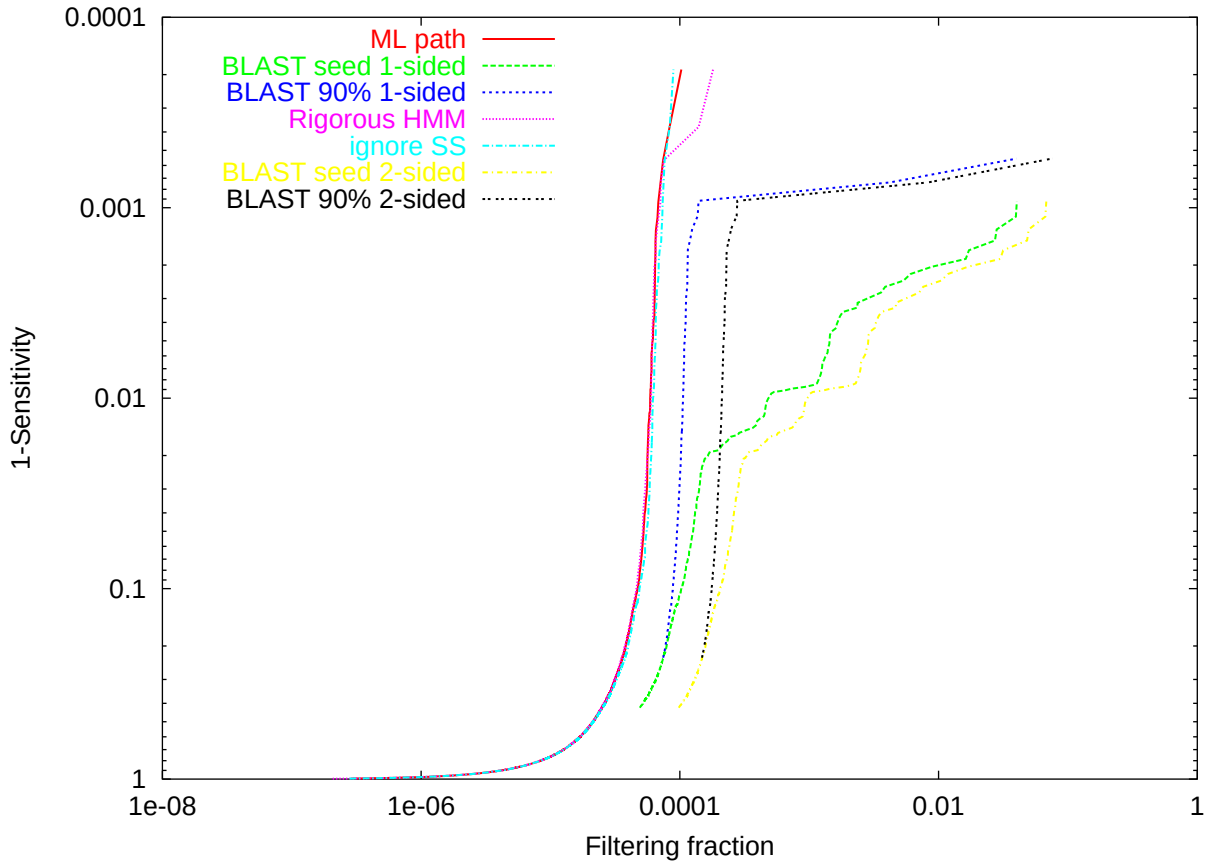

Figure 2: ROC-like curve: RF00001 (log-scale sensitivity)

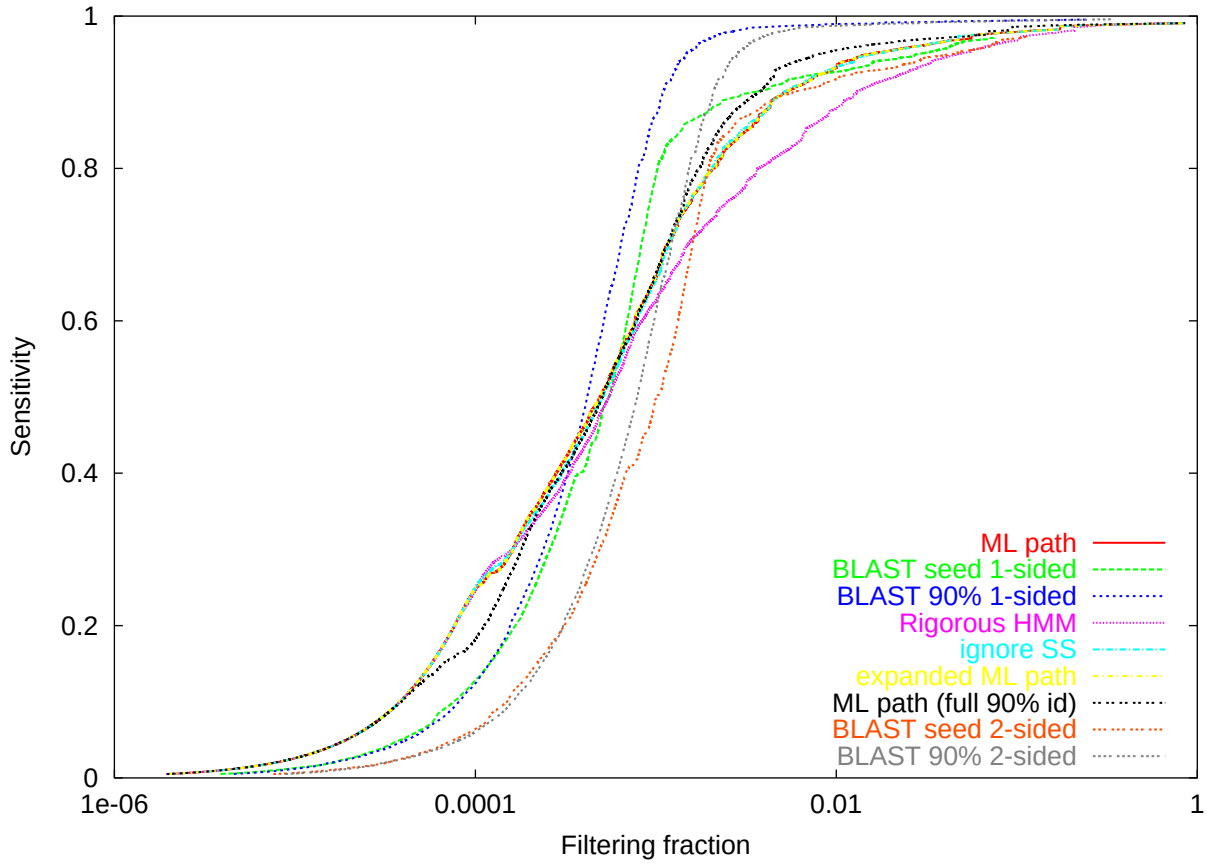

Figure 3: ROC-like curve: RF00005

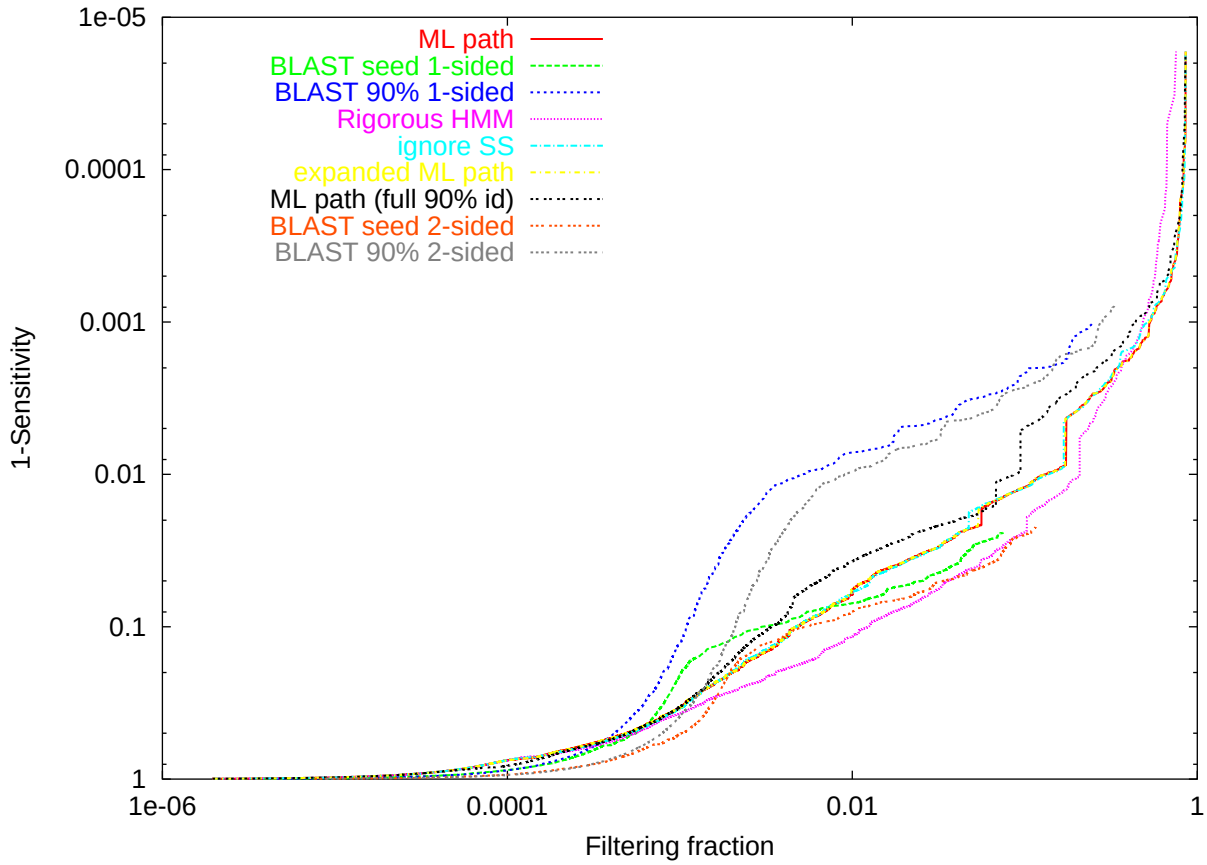

Figure 4: ROC-like curve: RF00005 (log-scale sensitivity)

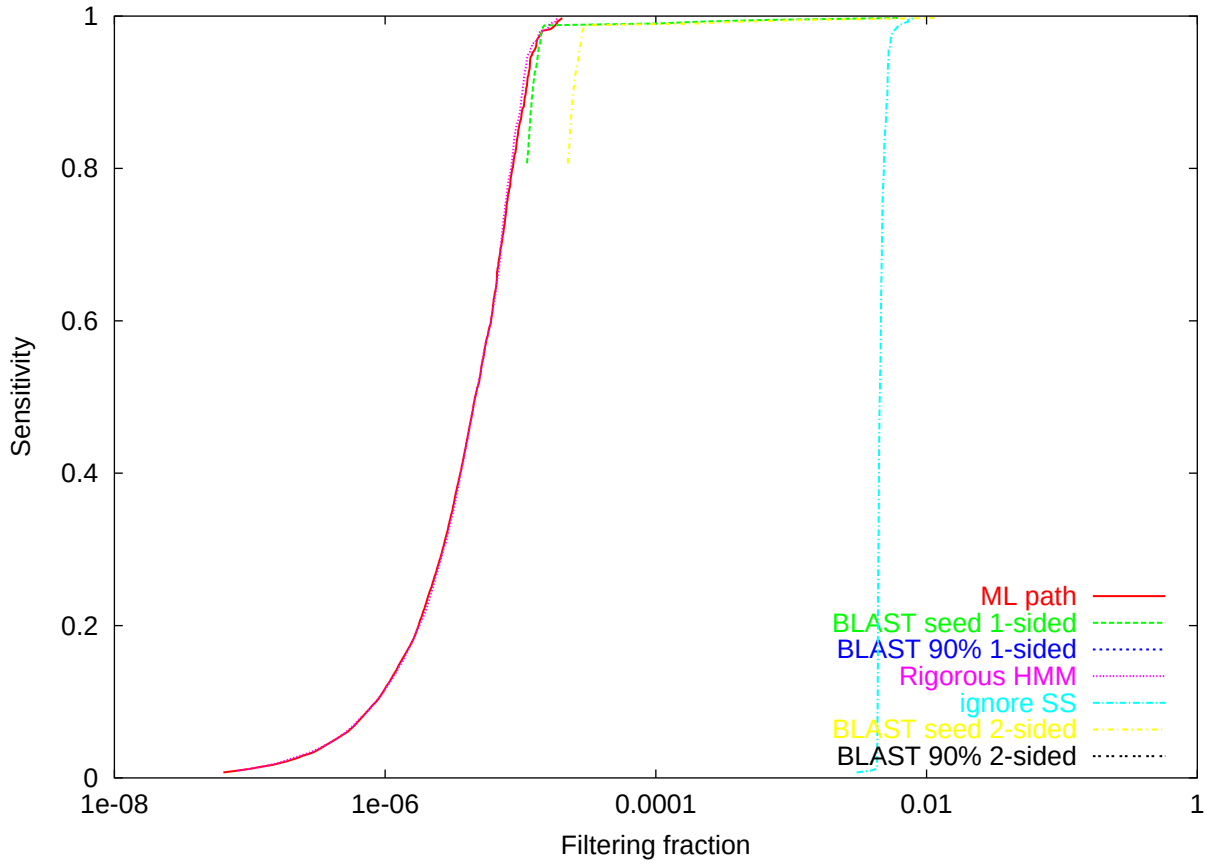

Figure 5: ROC-like curve: RF00010

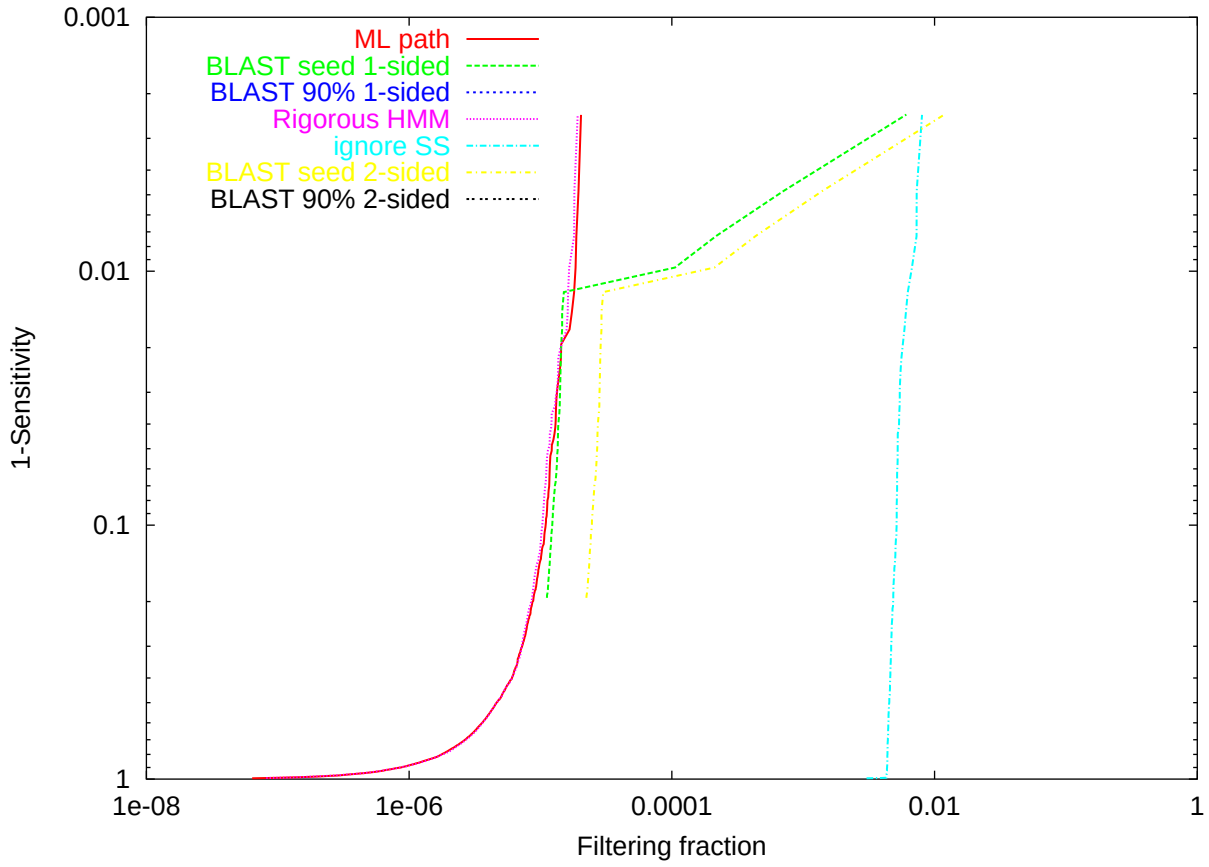

Figure 6: ROC-like curve: RF00010 (log-scale sensitivity)

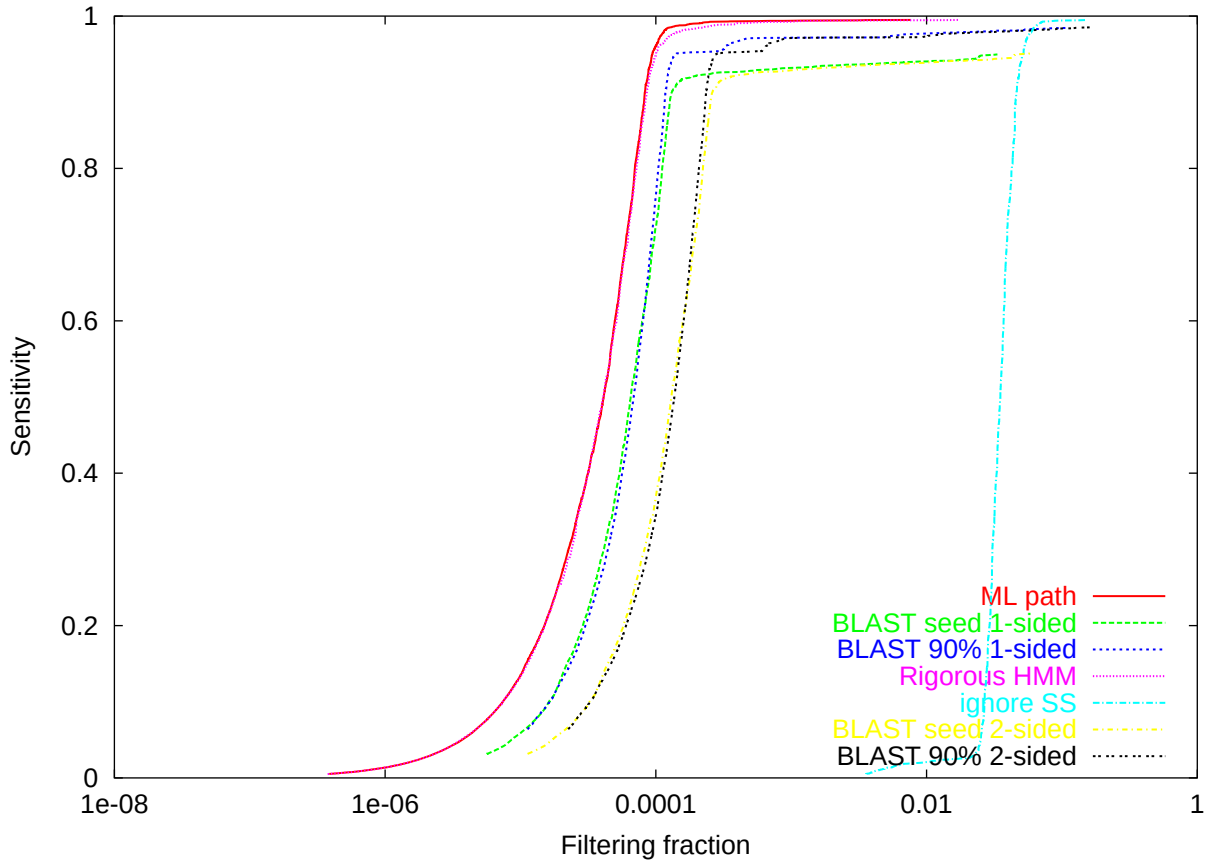

Figure 7: ROC-like curve: RF00029

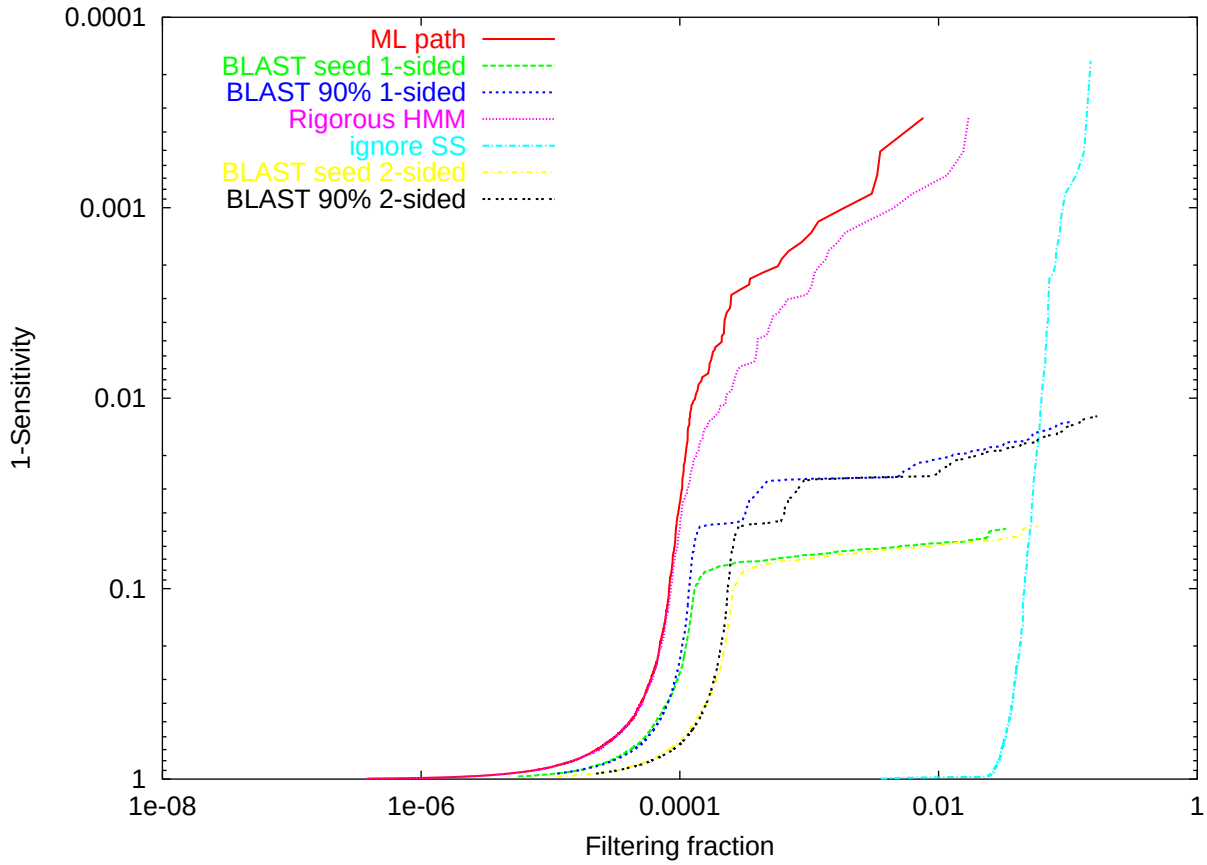

Figure 8: ROC-like curve: RF00029 (log-scale sensitivity)

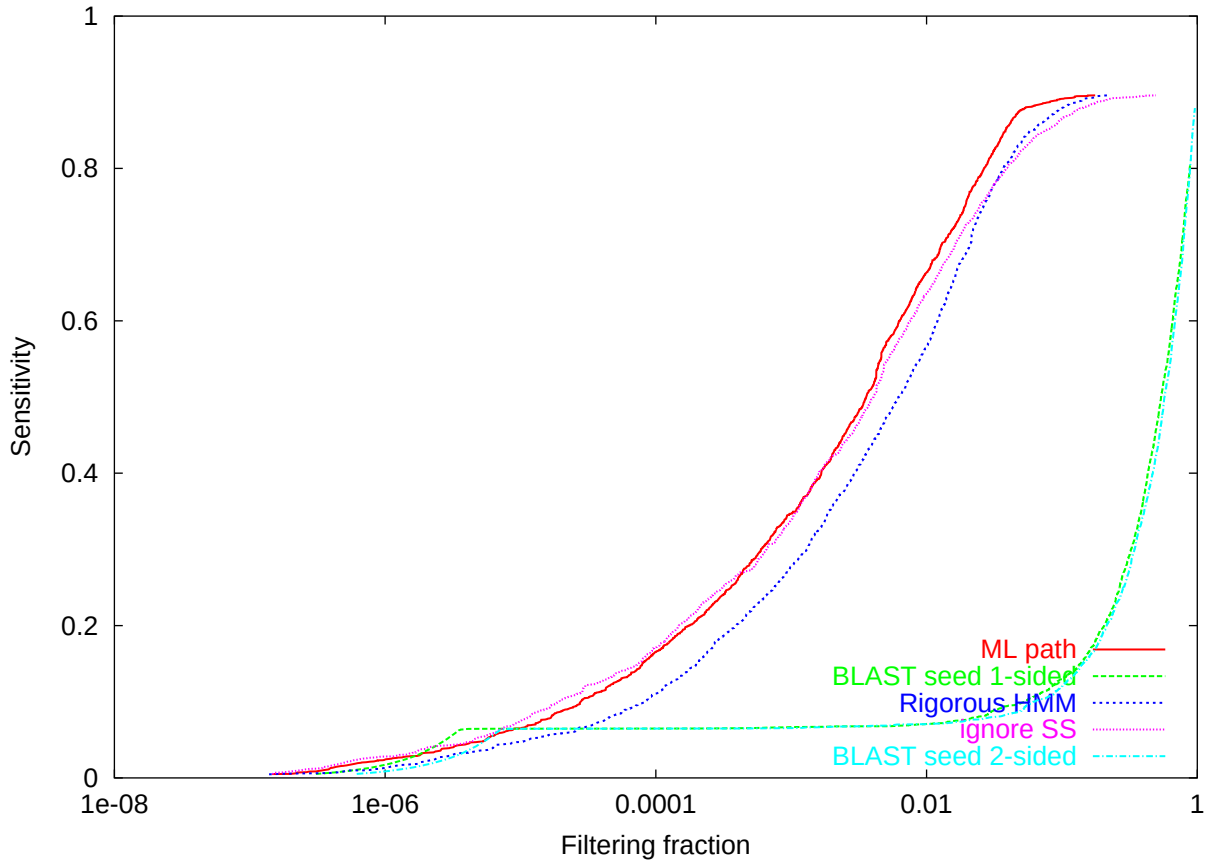

Figure 9: ROC-like curve: RF00031

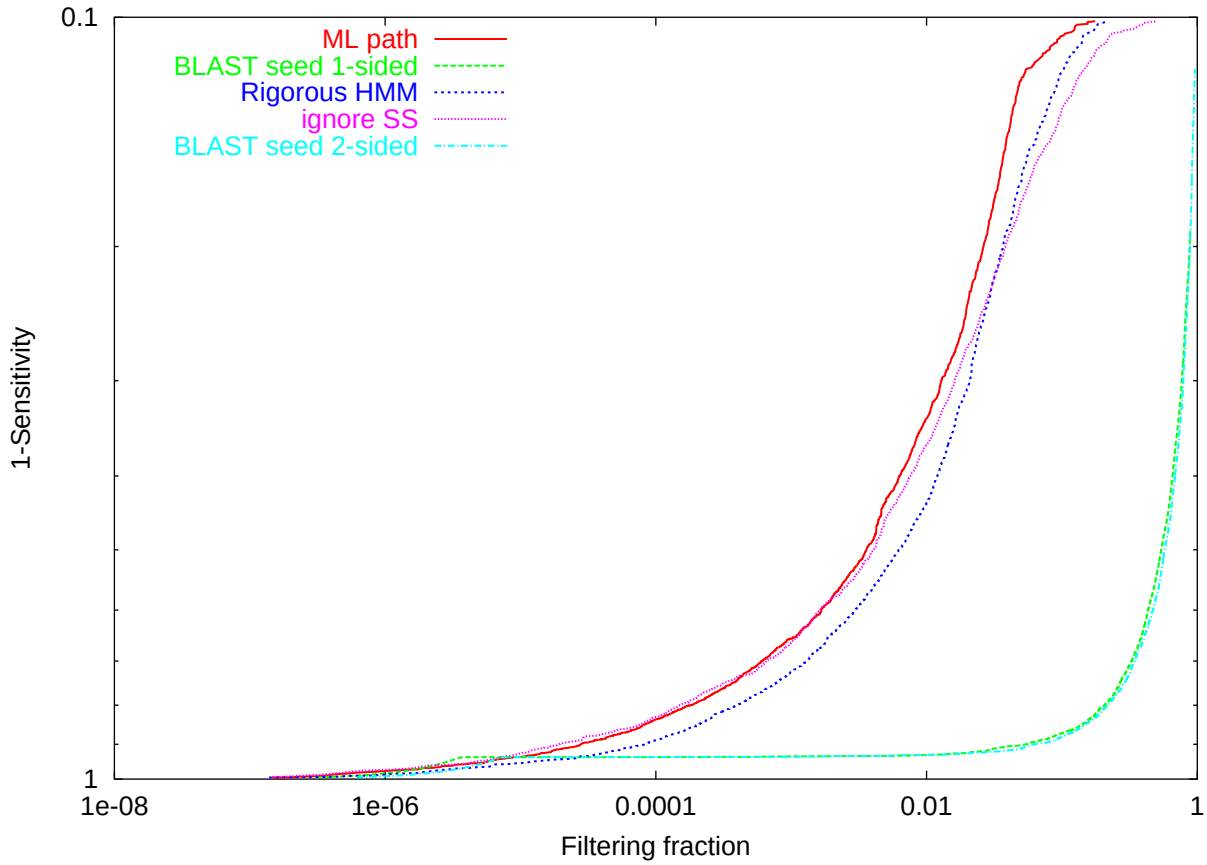

Figure 10: ROC-like curve: RF00031 (log-scale sensitivity)

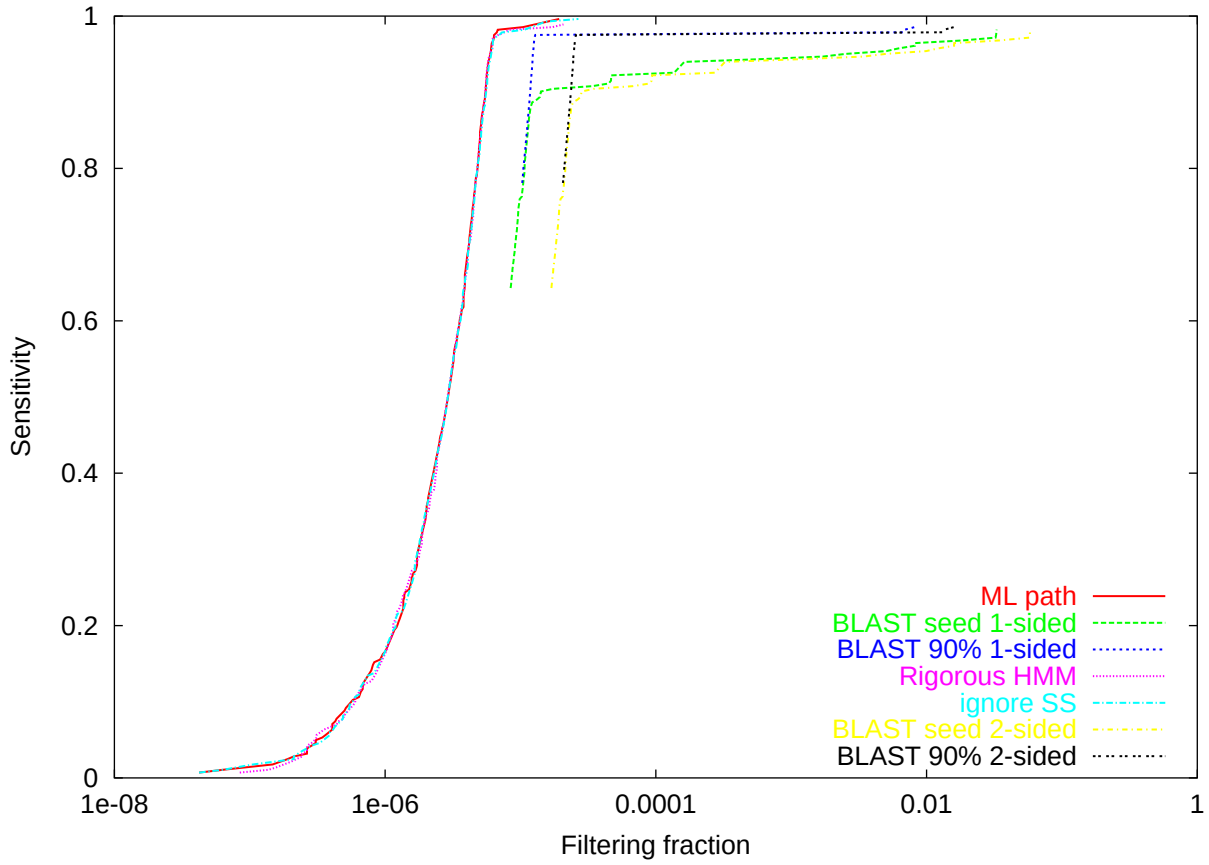

Figure 11: ROC-like curve: RF00059

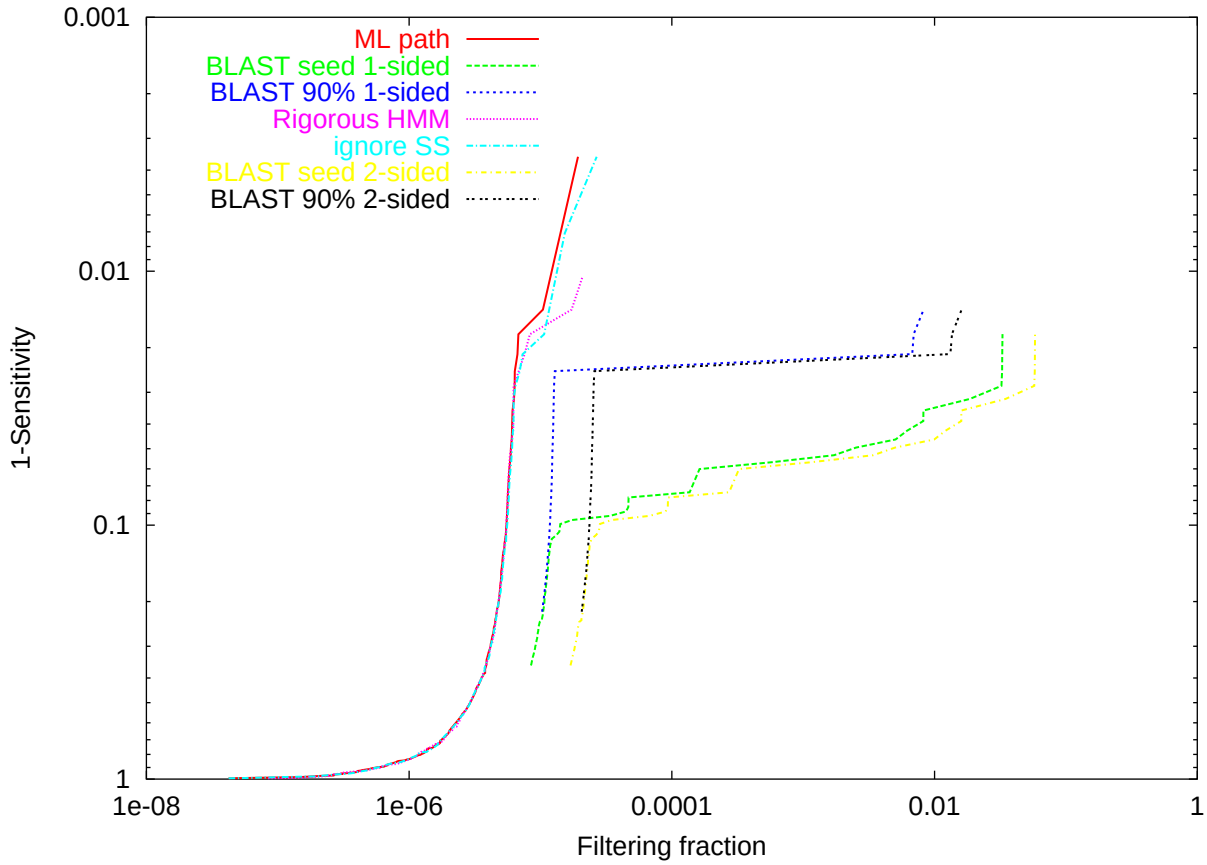

Figure 12: ROC-like curve: RF00059 (log-scale sensitivity)

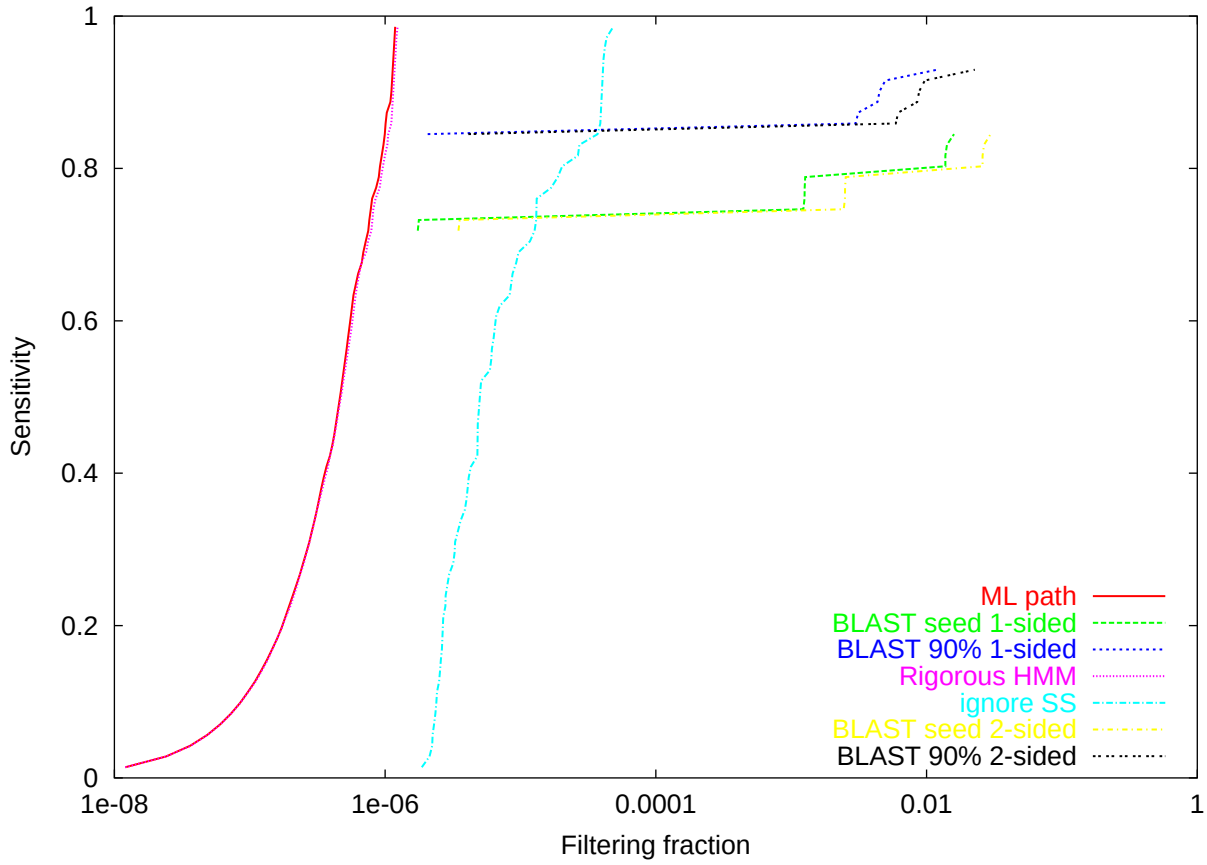

Figure 13: ROC-like curve: RF00168

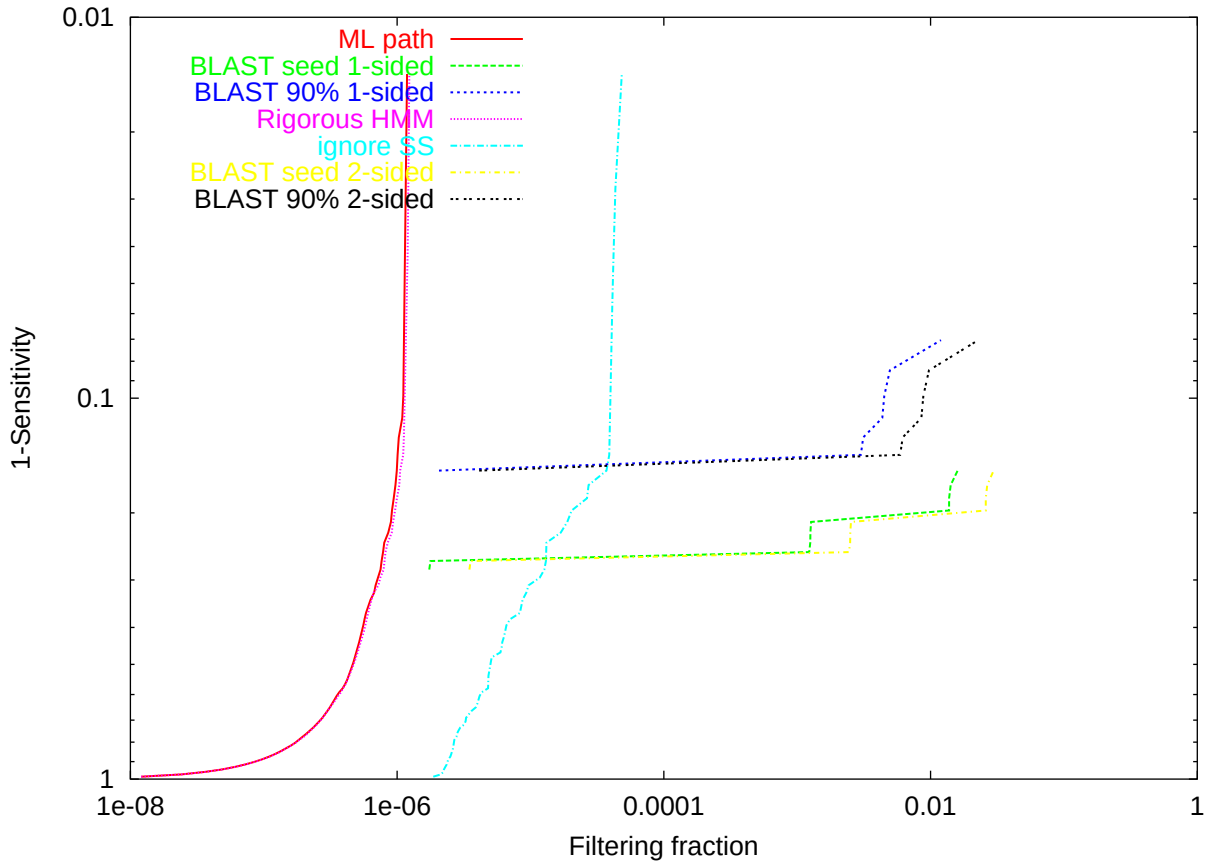

Figure 14: ROC-like curve: RF00168 (log-scale sensitivity)

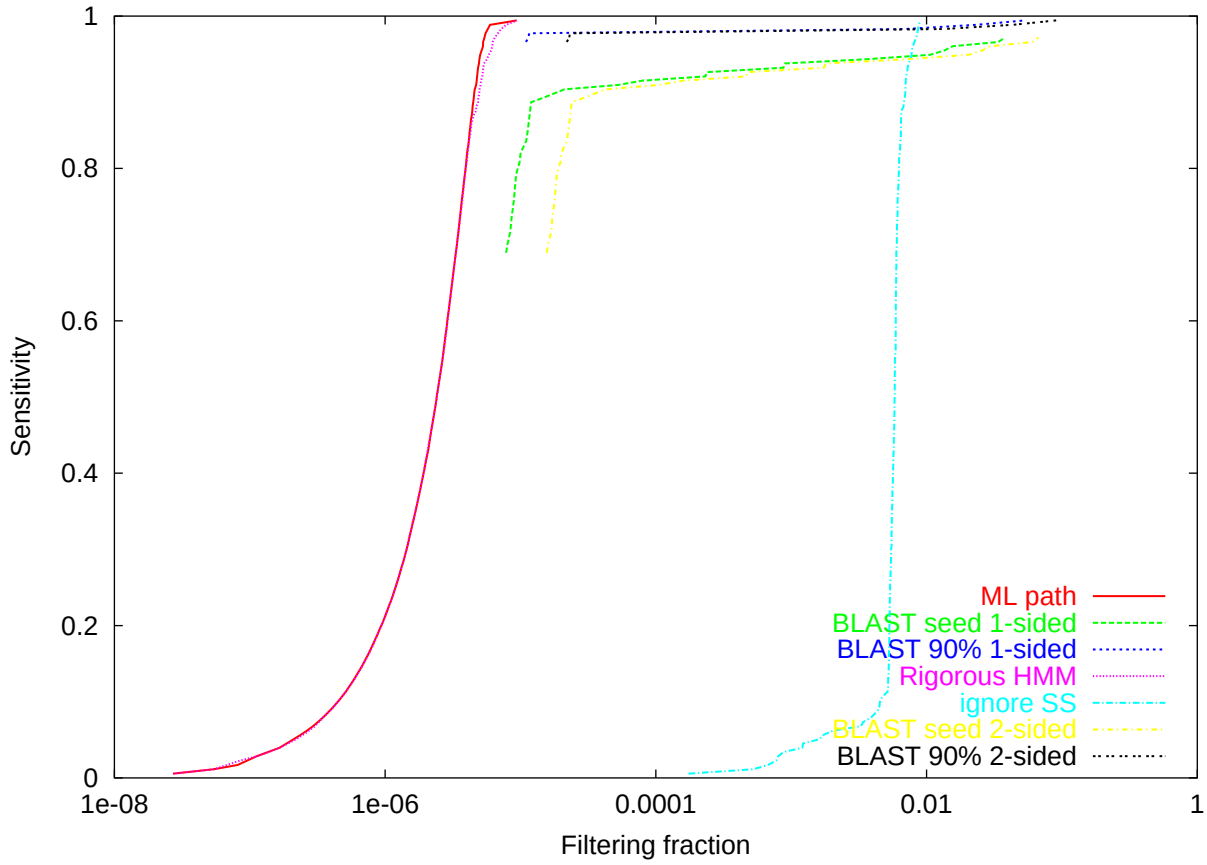

Figure 15: ROC-like curve: RF00174

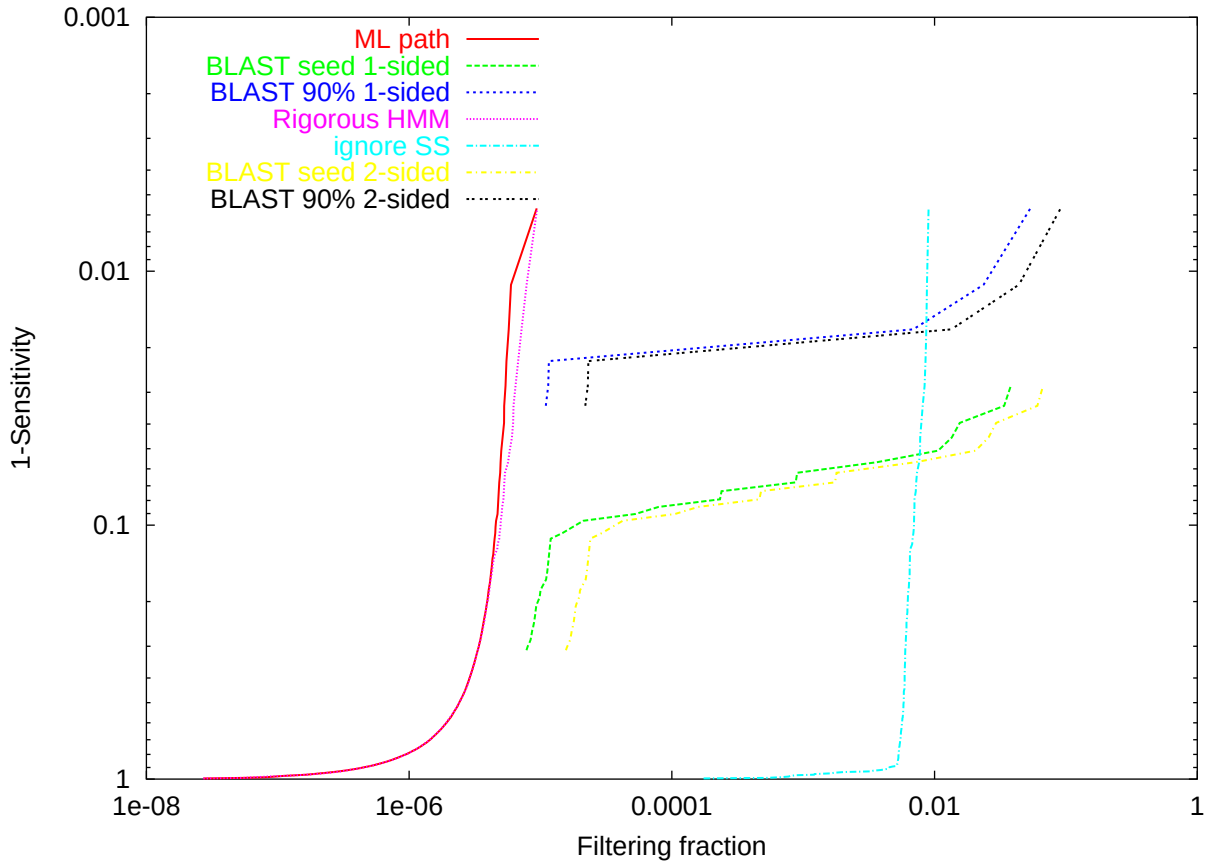

Figure 16: ROC-like curve: RF00174 (log-scale sensitivity)