HMMER User's Guide

Biological sequence analysis using profile hidden Markov models

`http://hmmer.org/`
Version 3.1b2; February 2015

Sean R. Eddy, Travis J. Wheeler
and the HMMER development team

Contents

1 Introduction

HMMER is used to search sequence databases for homologs of protein or DNA sequences, and to make sequence alignments. HMMER can be used to search sequence databases with single query sequences but it becomes particularly powerful when the query is an alignment of multiple instances of a sequence family. HMMER makes a *profile* of the query that assigns a position-specific scoring system for substitutions, insertions, and deletions. HMMER profiles are probabilistic models called "profile hidden Markov models" (profile HMMs) (Krogh et al., 1994; Eddy, 1998; Durbin et al., 1998).

Compared to BLAST, FASTA, and other sequence alignment and database search tools based on older scoring methodology, HMMER aims to be significantly more accurate and more able to detect remote homologs, because of the strength of its underlying probability models. In the past, this strength came at a significant computational cost, with profile HMM implementations running about 100x slower than comparable BLAST searches for protein search, and about 1000x slower than BLAST searches for DNA search. With HMMER3.1, HMMER is now essentially as fast as BLAST for protein search, and roughly 5-10x slower than BLAST in DNA search[1].

How to avoid reading this manual

We hate reading documentation. If you're like us, you're sitting here thinking, 117 pages of documentation, you must be joking! I just want to know that the software compiles, runs, and gives apparently useful results, before I read some 117 exhausting pages of someone's documentation. For cynics that have seen one too many software packages that don't work:

- Follow the quick installation instructions on page 13. An automated test suite is included, so you will know immediately if something went wrong.[2]

- Go to the tutorial section on page 18, which walks you through some examples of using HMMER on real data.

Everything else, you can come back and read later.

How to avoid using this software (links to similar software)

Several implementations of profile HMM methods and related position-specific scoring matrix methods are available.

Software	URL
HMMER	`http://hmmer.org/`
SAM	`http://www.cse.ucsc.edu/research/compbio/sam.html`
PSI-BLAST	`http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/psi1.htm`
PFTOOLS	`http://www.isrec.isb-sib.ch/profile/profile.html`

The UC Santa Cruz SAM software is the closest relative of HMMER.

What profile HMMs are

Profile HMMs are statistical models of multiple sequence alignments, or even of single sequences. They capture position-specific information about how conserved each column of the alignment is, and which residues[3] are likely. Anders Krogh, David Haussler, and co-workers at UC Santa Cruz introduced profile

---

[1]NCBI blastn with `--wordsize 7`; default wordsize of 11 is ∼10x faster, but much less sensitive.

[2]Nothing should go wrong.

[3]In some circles, "residue" is used to refer specifically in reference to amino acids, but here we allow the term to alternately refer to nucleotides.

HMMs to computational biology (Krogh et al., 1994), adopting HMM techniques which have been used for years in speech recognition. HMMs had been used in biology before the Krogh/Haussler work, notably by Gary Churchill (Churchill, 1989), but the Krogh paper had a dramatic impact because HMM technology was so well-suited to the popular "profile" methods for searching databases using multiple sequence alignments instead of single query sequences.

"Profiles" had been introduced by Gribskov and colleagues (Gribskov et al., 1987, 1990), and several other groups introduced similar approaches at about the same time, such as "flexible patterns" (Barton, 1990), and "templates"(Bashford et al., 1987; Taylor, 1986). The term "profile" has stuck.[4] All profile methods (including PSI-BLAST (Altschul et al., 1997)) are more or less statistical descriptions of the consensus of a multiple sequence alignment. They use *position-specific* scores for amino acids or nucleotides (residues) and position specific penalties for opening and extending an insertion or deletion. Traditional pairwise alignment (for example, BLAST (Altschul et al., 1990), FASTA (Pearson and Lipman, 1988), or the Smith/Waterman algorithm (Smith and Waterman, 1981)) uses position-*independent* scoring parameters. This property of profiles captures important information about the degree of conservation at various positions in the multiple alignment, and the varying degree to which gaps and insertions are permitted.

The advantage of using HMMs is that HMMs have a formal probabilistic basis. We use probability theory to guide how all the scoring parameters should be set. Though this might sound like a purely academic issue, this probabilistic basis lets us do things that more heuristic methods cannot do easily. One of the most important is that HMMs have a consistent theory for setting position-specific gap and insertion scores. The methods are consistent and therefore highly automatable, allowing us to make libraries of hundreds of profile HMMs and apply them on a very large scale to whole genome analysis. One such database of protein domain models is Pfam (Sonnhammer et al., 1997; Finn et al., 2010), which is a significant part of the Interpro protein domain annotation system (Mulder et al., 2003). The construction and use of Pfam is tightly tied to the HMMER software package.

Profile HMMs do have important limitations. One is that HMMs do not capture any higher-order correlations. An HMM assumes that the identity of a particular position is independent of the identity of all other positions.[5] Profile HMMs are often not good models of structural RNAs, for instance, because an HMM cannot describe base pairs.

Applications of profile HMMs

HMMER can be used to replace BLASTP and PSI-BLAST for searching protein databases with single query sequences. HMMER includes two programs for searching protein databases with single query sequences: `phmmer` and `jackhmmer`, where `jackhmmer` is an iterative search akin to PSI-BLAST.

Another application of HMMER is when you are working on a sequence family, and you have carefully constructed a multiple sequence alignment. Your family, like most protein (or DNA) families, has a number of strongly (but not absolutely) conserved key amino acids (or nucleotides), separated by characteristic spacing. You wonder if there are more members of your family in the sequence databases, but the family is so evolutionarily diverse, a BLAST search with any individual sequence doesn't even find the rest of the sequences you already know about. You're sure there are some distantly related sequences in the noise. You spend many pleasant evenings scanning weak BLAST alignments by eye to find ones with the right key residues are in the right places, but you wish there was a computer program that did this a little better.

Another application is the automated annotation of the domain structure of proteins. Large databases of curated alignments and HMMER models of known domains are available, including Pfam (Finn et al., 2010) and SMART (Letunic et al., 2006) in the Interpro database consortium (Mulder et al., 2003). (Many "top ten protein domains" lists, a standard table in genome analysis papers, rely heavily on HMMER annotation.) Say you have a new sequence that, according to a BLAST analysis, shows a slew of hits to receptor

---

[4]There has been agitation in some quarters to call all such models "position specific scoring matrices", PSSMs, but certain small nocturnal North American marsupials have a prior claim on the name.

[5]This is not strictly true. There is a subtle difference between an HMM's state path (a first order Markov chain) and the sequence described by an HMM (generated from the state path by independent emissions of symbols at each state).

tyrosine kinases. Before you decide to call your sequence an RTK homologue, you suspiciously recall that RTK's are, like many proteins, composed of multiple functional domains, and these domains are often found promiscuously in proteins with a wide variety of functions. Is your sequence really an RTK? Or is it a novel sequence that just happens to have a protein kinase catalytic domain or fibronectin type III domain?

Another application is the automated construction and maintenance of large multiple alignment databases. It is useful to organize sequences into evolutionarily related families. But there are thousands of sequence families, some of which comprise tens of thousands of sequences – and the primary sequence databases continue to double every year or two. This is a hopeless task for manual curation; but on the other hand, manual curation is still necessary for high-quality, biologically relevant multiple alignments. Databases like Pfam (Finn et al., 2010) and Dfam (Wheeler et al., 2013) are constructed by distinguishing between a stable curated "seed" alignment of a small number of representative sequences, and "full" alignments of all detectable homologs; HMMER is used to make a model of the seed, search the database for homologs, and can automatically produce the full alignment by aligning every sequence to the seed consensus.

You may find other applications as well. Using hidden Markov models to make a linear consensus model of a bunch of related strings is a pretty generic problem, and not just in biological sequence analysis. HMMER3 has already been used to model mouse song [Elena Rivas, personal communication] and in the past HMMER has reportedly been used to model music, speech, and even automobile engine telemetry. If you use it for something particularly strange, we'd be curious to hear about it – but Sean never, ever wants to see these error messages showing up on the console of his Saab.

Design goals of HMMER3

In the past, profile HMM methods were considered to be too computationally expensive, and BLAST has remained the main workhorse of sequence similarity searching. The main objective of the HMMER3 project (begun in 2004) is to combine the power of probabilistic inference with high computational speed. We aim to upgrade some of molecular biology's most important sequence analysis applications to use more powerful statistical inference engines, without sacrificing computational performance.

Specifically, HMMER3 has three main design features that *in combination* distinguish it from previous tools:

Explicit representation of alignment uncertainty. Most sequence alignment analysis tools report only a single best-scoring alignment. However, sequence alignments are uncertain, and the more distantly related sequences are, the more uncertain alignments become. HMMER3 calculates complete posterior alignment ensembles rather than single optimal alignments. Posterior ensembles get used for a variety of useful inferences involving alignment uncertainty. For example, any HMMER sequence alignment is accompanied by posterior probability annotation, representing the degree of confidence in each individual aligned residue.

Sequence scores, not alignment scores. Alignment uncertainty has an important impact on the power of sequence database searches. It's precisely the most remote homologs – the most difficult to identify and potentially most interesting sequences – where alignment uncertainty is greatest, and where the statistical approximation inherent in scoring just a single best alignment breaks down the most. To maximize power to discriminate true homologs from nonhomologs in a database search, statistical inference theory says you ought to be scoring sequences by integrating over alignment uncertainty, not just scoring the single best alignment. HMMER3's log-odds scores are sequence scores, not just optimal alignment scores; they are integrated over the posterior alignment ensemble.

Speed. A major limitation of previous profile HMM implementations (including HMMER2) was their slow performance. HMMER3 implements a heuristic acceleration algorithm. For most protein queries, it's about as fast as BLAST, while for DNA queries it's typically less than 10x slower than sensitive settings for BLAST.

Individually, none of these points is new. As far as alignment ensembles and sequence scores go, pretty much the whole reason why hidden Markov models are so theoretically attractive for sequence analysis is that they are good probabilistic models for explicitly dealing with alignment uncertainty. The SAM profile HMM software from UC Santa Cruz has always used full probabilistic inference (the HMM Forward and Backward algorithms) as opposed to optimal alignment scores (the HMM Viterbi algorithm). HMMER2 had the full HMM inference algorithms available as command-line options, but used Viterbi alignment by default, in part for speed reasons. Calculating alignment ensembles is even more computationally intensive than calculating single optimal alignments.

One reason why it's been hard to deploy sequence scores for practical large-scale use is that we haven't known how to accurately calculate the statistical significance of a log-odds score that's been integrated over alignment uncertainty. Accurate statistical significance estimates are essential when one is trying to discriminate homologs from millions of unrelated sequences in a large sequence database search. The statistical significance of optimal alignment scores can be calculated by Karlin/Altschul statistics (Karlin and Altschul, 1990, 1993). Karlin/Altschul statistics are one of the most important and fundamental advances introduced by BLAST. However, this theory doesn't apply to integrated log-odds sequence scores (HMM "Forward scores"). The statistical significance (expectation values, or E-values) of HMMER3 sequence scores is determined by using recent theoretical conjectures about the statistical properties of integrated log-odds scores which have been supported by numerical simulation experiments (Eddy, 2008).

And as far as speed goes, there's really nothing new about HMMER3's speed either. Besides Karlin/Altschul statistics, the main reason BLAST has been so useful is that it's so fast. Our design goal in HMMER3 was to achieve rough speed parity between BLAST and more formal and powerful HMM-based methods. The acceleration algorithm in HMMER3 is a new heuristic. It seems likely to be more sensitive than BLAST's heuristics on theoretical grounds. It certainly benchmarks that way in practice, at least in our hands. Additionally, it's very well suited to modern hardware architectures. We expect to be able to take good advantage of GPUs and other parallel processing environments in the near future.

What's new in HMMER3.1

DNA sequence comparison. HMMER now includes tools that are specifically designed for DNA/DNA comparison: `nhmmer` and `nhmmscan`. The most notable improvement over using HMMER3's tools is the ability to search long (e.g. chromosome length) target sequences.

More sequence input formats. HMMER now handles a wide variety of input sequence file formats, both aligned (Stockholm, Aligned FASTA, Clustal, NCBI PSI-BLAST, PHYLIP, Selex, UCSC SAM A2M) and unaligned (FASTA, EMBL, Genbank), usually with autodetection.

MSV stage of HMMER acceleration pipeline now even faster. Bjarne Knudsen, Chief Scientific Officer of CLC bio in Denmark, contributed an important optimization of the MSV filter (the first stage in the accelerated "filter pipeline") that increases overall HMMER3 speed by about two-fold. This speed improvement has no impact on sensitivity.

Faster DNA search. This beta release includes a new option for increasing the speed of searching DNA sequence files. A binary file is created in a preprocessing step, using the tool `makehmmerdb`. The tool `nhmmer` then utilizes this binary format to support a fast new seed-finding stage. Using default settings in `nhmmer`, we have observed a roughly 10-fold acceleration with small loss of sensitivity on benchmarks. (This method has been extensively tested, but should still be treated as somewhat experimental.)

What's still missing in HMMER3.1

Even though this is a stable public release that we consider suitable for large-scale production work (genome annotation, Pfam analysis, whatever), at the same time, HMMER remains a work in progress. We think the codebase is a suitable foundation for development of a number of significantly improved approaches to classically important problems in sequence analysis. Some of the more important "holes" for us are:

Translated comparisons.   We'd of course love to have the HMM equivalents of BLASTX, TBLASTN, and TBLASTX. They'll come.

Profile/profile comparison.   A number of pioneering papers and software packages have demonstrated the power of profile/profile comparison for even more sensitive remote homology detection. We will aim to develop profile HMM methods in HMMER with improved detection power, and at HMMER3 speed.

We also have some technical and usability issues we will be addressing Real Soon Now:

More alignment modes.   HMMER3 *only* does local alignment. HMMER2 also could do glocal alignment (align a complete model to a subsequence of the target) and global alignment (align a complete model to a complete target sequence). The E-value statistics of glocal and global alignment remain poorly understood. HMMER3 relies on accurate significance statistics, far more so than HMMER2 did, because HMMER3's acceleration pipeline works by filtering out sequences with poor P-values.

More speed.   Even on x86 platforms, HMMER3's acceleration algorithms are still on a nicely sloping bit of their asymptotic optimization curve. We still think we can accelerate the code by another two-fold or so, perhaps more for DNA search. Additionally, for a small number of HMMs ($< 1\%$ of Pfam models), the acceleration core is performing relatively poorly, for reasons we pretty much understand (having to do with biased composition; most of these pesky models are hydrophobic membrane proteins), but which are non-trivial to work around. This'll produce an annoying behavior that you may notice: if you look systematically, sometimes you'll see a model that runs at something more like HMMER2 speed, 100x or so slower than an average query. This, needless to say, Will Be Fixed.

More processor support.   One of the attractive features of the HMMER3 "MSV" acceleration algorithm is that it is a very tight and efficient piece of code, which ought to be able to take advantage of recent advances in using massively parallel GPUs (graphics processing units), and other specialized processors such as the Cell processor, or FPGAs. We have prototype work going on in a variety of processors, but none of this is far along as yet. But this work (combined with the parallelization) is partly why we expect to wring significantly more speed out of HMMER in the future.

How to learn more about profile HMMs

Cryptogenomicon (`http://cryptogenomicon.org/`) is a blog where Sean will be talking about issues as they arise in HMMER3, and where you can comment or follow the discussion.

Reviews of the profile HMM literature have been written by Sean (Eddy, 1996, 1998) and by Anders Krogh (Krogh, 1998).

For details on how profile HMMs and probabilistic models are used in computational biology, see the pioneering 1994 paper from Krogh et al. (Krogh et al., 1994) or our book *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Durbin et al., 1998).

For details on many aspects of the inner workings of HMMER3, see recent papers by Sean (Eddy, 2009, 2011).

To learn more about HMMs from the perspective of the speech recognition community, an excellent tutorial introduction has been written by Rabiner (Rabiner, 1989).

> ▷ **How do I cite HMMER?** *There is still no "real" paper on HMMER. If you're writing for an enlightened (url-friendly) journal, the best reference is* `http://hmmer.org/`. *If you must use a paper reference, the best one to use is Sean's 1998 profile HMM review (Eddy, 1998).*

2 Installation

Quick installation instructions

Download `hmmer-3.1b2.tar.gz` from `http://hmmer.org/`; unpack it:

> **wget ftp://selab.janelia.org/pub/software/hmmer3/3.1b2/hmmer-3.1b2.tar.gz**
> **tar xf hmmer-3.1b2.tar.gz**
> **cd hmmer-3.1b1**

If you have downloaded a variant "with binaries", the pre-compiled binaries are found in the directory `./binaries`. You may copy these into a directory of your choosing, or simply run them from here (for example, by adding this directory to your `PATH` variable).

If you have downloaded the HMMER source code, you'll need to compile the software with configure and make:

> **./configure**
> **make**

To compile and run a test suite to make sure all is well, you can optionally do:

> **make check**

All these tests should pass.

You don't have to install HMMER programs to run them. The newly compiled binaries are now in the `src` directory. You can run them from there. To install the programs and man pages somewhere on your system, do:

> **make install**

By default, programs are installed in `/usr/local/bin` and man pages in `/usr/local/share/man/man1/`. You can change the `/usr/local` prefix to any directory you want using the `./configure --prefix` option, as in `./configure --prefix /the/directory/you/want`.

Optionally, you can install the Easel library package as well, including its various "miniapplications", in addition to its library and header files. We don't do this by default, in case you already have a copy of Easel separately installed:

> **cd easel; make install**

That's it. You can keep reading if you want to know more about customizing a HMMER installation, or you can skip ahead to the next chapter, the tutorial.

System requirements

Operating system: HMMER is designed to run on POSIX-compatible platforms, including UNIX, Linux and MacOS/X. The POSIX standard essentially includes all operating systems except Microsoft Windows.[1] We have tested most extensively on Linux and on MacOS/X, because these are the machines we develop on.

Processor: HMMER depends on vector parallelization methods that are supported on most modern processors. H3 requires either an x86-compatible (IA32, IA64, or Intel64) processor that supports the SSE2 vector instruction set, or a PowerPC processor that supports the Altivec/VMX instruction set. SSE2 is supported on Intel processors from Pentium 4 on, and AMD processors from K8 (Athlon 64) on; we believe this includes almost all Intel processors since 2000 and AMD processors since 2003. Altivec/VMX is supported on Motorola G4, IBM G5, and IBM PowerPC processors starting with the Power6, which we believe includes almost all PowerPC-based desktop systems since 1999 and servers since 2007.

If your platform does not support one of these vector instruction sets, the configure script will revert to an unoptimized implementation called the "dummy" implementation. **The dummy implementation is two**

---

[1]There are add-on products available for making Windows more POSIX-compliant and more compatible with GNU-ish configures and builds. One such product is Cygwin, `http:www.cygwin.com`, which is freely available. Although we do not test on Windows platforms, we understand HMMER builds fine in a Cygwin environment on Windows.

orders of magnitude slower. It will enable you to see H3's scientific features on a much wider range of processors, but is not suited for real production work.

We do aim to be portable to all modern processors. The acceleration algorithms are designed to be portable despite their use of specialized SIMD vector instructions. We hope to add support for the Sun SPARC VIS instruction set, for example. We believe that the code will be able to take advantage of GP-GPUs and FPGAs in the future.

Compiler: The source code is C conforming to POSIX and ANSI C99 standards. It should compile with any ANSI C99 compliant compiler, including the GNU C compiler `gcc`. We test the code using both the `gcc` and `icc` compilers. We find that `icc` produces somewhat faster code at present.

Libraries and other installation requirements: HMMER includes a software library called Easel, which it will automatically compile during its installation process. By default, HMMER3 does not require any additional libraries to be installed by you, other than standard ANSI C99 libraries that should already be present on a system that can compile C code. Bundling Easel instead of making it a separate installation requirement is a deliberate design decision to simplify the installation process.[2]

Configuration and compilation use several UNIX utilities. Although these utilities are available on all UNIX/Linux/MacOS systems, old versions may not support all the features the `./configure` script and Makefiles are hoping to find. We aim to build on anything, even old Ebay'ed junk, but if you have an old system, you may want to hedge your bets and install up-to-date versions of GNU tools such as GNU make and GNU grep.

Multithreaded parallelization for multicores is the default

The six search programs and `hmmbuild` support multicore parallelization using POSIX threads. By default, the configure script will identify whether your platform supports POSIX threads (almost all platforms do), and will automatically compile in multithreading support.

If you want to disable multithreading at compile time, recompile from source after giving the `--disable-threads` flag to `./configure`.

By default, our multithreaded programs will use all available cores on your machine. You can control the number of cores each HMMER process will use for computation with the `--cpu <x>` command line option or the `HMMER_NCPU` environment variable. Even with a single processing thread (`--cpu 1`), HMMER will devote a second execution thread to database input, resulting in possible speedup over serial execution.

If you specify `--cpu 0`, the program will run in serial-only mode, with no threads. This might be useful if you suspect something is awry with the threaded parallel implementation.

MPI parallelization for clusters is optional

MPI (Message Passing Interface) parallelization on clusters is now supported in hmmbuild and all search programs except nhmmer and nhmmscan. To use MPI, you first need to have an MPI library installed, such as OpenMPI (`www.open-mpi.org`). We use Intel MPI at Janelia.

MPI support is not enabled by default, and it is not compiled into the precompiled binaries that we supply with HMMER. To enable MPI support at compile time, give the `--enable-mpi` option to the `./configure` command.

To use MPI parallelization, each program that has an MPI-parallel mode has an `--mpi` command line option. This option activates a master/worker parallelization mode. (Without the `--mpi` option, if you run a program under `mpirun` on N nodes, you'll be running N independent duplicate commands, not a single MPI-enabled command. Don't do that.)

---

[2]If you install more than one package that uses the Easel library, it may become an annoyance; you'll have multiple instantiations of Easel lying around. The Easel API is not yet stable enough to decouple it from the applications that use it, like HMMER and Infernal.

The MPI implementation for `hmmbuild` scales well up to hundreds of processors, and `hmmsearch` scales all right. The other search programs (`hmmscan`, `phmmer`, and `jackhmmer`) scale poorly, and probably shouldn't be used on more than tens of processors at most. Improving MPI scaling is one of our goals.

Using build directories

The configuration and compilation process from source supports using separate build directories, using the GNU-standard VPATH mechanism. This allows you to maintain separate builds for different processors or with different configuration/compilation options. All you have to do is run the configure script from the directory you want to be the root of your build directory. For example:

```
> mkdir my-hmmer-build
> cd my-hmmer-build
> /path/to/hmmer/configure
> make
```

This assumes you have a `make` that supports VPATH. If your system's `make` does not, you can always install GNU make.

Makefile targets

all Builds everything. Same as just saying `make`.

check Runs automated test suites in both HMMER and the Easel library.

clean Removes all files generated by compilation (by `make`). Configuration (files generated by `./configure`) is preserved.

distclean Removes all files generated by configuration (by `./configure`) and by compilation (by `make`).

Note that if you want to make a new configuration (for example, to try an MPI version by `./configure --enable-mpi; make`) you should do a `make distclean` (rather than a `make clean`), to be sure old configuration files aren't used accidentally.

Why is the output of 'make' so clean?

Because we're hiding what's really going on with the compilation with a pretty wrapper. If you want to see what the command lines really look like, in all their ugly glory, pass a `V=1` option (V for "verbose") to `make`, as in:

```
> make V=1
```

What gets installed by 'make install', and where?

HMMER's 'make install' generally follows the GNU Coding Standards and the Filesystem Hierarchy Standard. The top-level Makefile has variables that specify five directories where `make install` will install things:

Variable	What
`bindir`	All HMMER programs
`libdir`	`libhmmer.a`
`includedir`	`hmmer.h`
`man1dir`	All HMMER man pages
`pdfdir`	`Userguide.pdf`

These variables are constructed from some other variables, in accordance with the GNU Coding Standards. All of these variables are at the top of the top-level Makefile. Their defaults are as follows:

15

Variable	Default
prefix	/usr/local
exec_prefix	$prefix
bindir	$exec_prefix/bin
libdir	$exec_prefix/lib
includedir	$prefix/include
datarootdir	$prefix/share
mandir	$datarootdir/man
man1dir	$mandir/man1

The best way to change these defaults is when you use `./configure`, and the most important variable to consider changing is `--prefix`. For example, if you want to install HMMER in a directory hierarchy all of its own, you might want to do something like:

> **> ./configure --prefix /usr/local/hmmer**

That would keep HMMER out of your system-wide directories like `/usr/local/bin`, which might be desirable. Of course, if you do it that way, you'd also want to add `/usr/local/hmmer/bin` to your `$PATH`, `/usr/local/hmmer/share/man` to your `$MANPATH`, etc.

These variables only affect `make install`. HMMER executables have no pathnames compiled into them.

Staged installations in a buildroot, for a packaging system

HMMER's `make install` supports staged installations, accepting the traditional `DESTDIR` variable that packagers use to specify a buildroot. For example, you can do:

> **> make DESTDIR=/rpm/tmp/buildroot install**

Workarounds for some unusual configure/compilation problems

Configuration or compilation fails when trying to use a separate build directory. If you try to build in a build tree (other than the source tree) and you have any trouble in configuration or compilation, try just building in the source tree instead. Some `make` versions don't support the VPATH mechanism needed to use separate build trees. Another workaround is to install GNU make.

Configuration fails, complaining that the CFLAGS don't work. Our configure script uses an Autoconf macro, AX_CC_MAXOPT, that tries to guess good optimization flags for your compiler. In very rare cases, we've seen it guess wrong. You can always set CFLAGS yourself with something like:

> **> ./configure CFLAGS=-O**

Configuration fails, complaining "no acceptable grep could be found". We've seen this happen on our Sun Sparc/Solaris machine. It's a known issue in GNU autoconf. You can either install GNU grep, or you can insist to `./configure` that the Solaris grep (or whatever grep you have) is ok by explicitly setting GREP:

> **> ./configure GREP=/usr/xpg4/bin/grep**

Configuration warns that it's using the "dummy" implementation and H3 is going to be very slow. This is what you get if your system has a processor that we don't yet support with a fast vector-parallel implementation. We currently support Intel/AMD compatible processors and PowerPC compatible processors. H3 will revert to a portable but slow version on other processors.

Many 'make check' tests fail. We have one report of a system that failed to link multithread-capable system C libraries correctly, and instead linked to one or more serial-only libraries.[3] We've been unable to reproduce the problem here, and are not sure what could cause it – we optimistically believe it's a messed-up system instead of our fault. If it does happen, it screws all kinds of things up with the multithreaded implementation. A workaround is to shut threading off:

```
> ./configure --disable-threads
```

This will compile code won't parallelize across multiple cores, of course, but it will still work fine on a single processor at a time (and MPI, if you build with MPI enabled).

---

[3]The telltale phenotype of this failure is to configure with debugging flags on and recompile, run one of the failed unit test drivers (such as `easel/easel_utest`) yourself and let it dump core; and use a debugger to examine the stack trace in the core. If it's failed in `_errno_location()`, it's linked a non-thread-capable system C library.

3 Tutorial

Here's a tutorial walk-through of some small projects with HMMER. This should suffice to get you started on work of your own, and you can (at least temporarily) skip the rest of the Guide, such as all the nitty-gritty details of available command line options.

The programs in HMMER

Build models and align sequences (DNA or protein)

hmmbuild   Build a profile HMM from an input multiple alignment.
hmmalign   Make a multiple alignment of many sequences to a common profile HMM.

Search protein queries against protein database

phmmer   Search a single protein sequence against a protein sequence database. (BLASTP-like)
jackhmmer   Iteratively search a protein sequence against a protein sequence database. (PSIBLAST-like)
hmmsearch   Search a protein profile HMM against a protein sequence database.
hmmscan   Search a protein sequence against a protein profile HMM database.
hmmpgmd   Search daemon used for hmmer.org website.

Search DNA queries against DNA database

nhmmer   Search a DNA sequence, alignment, or profile HMM against a DNA sequence database. (BLASTN-like)
nhmmscan   Search a DNA sequence against a DNA profile HMM database.

Other utilities

alimask   Modify alignment file to mask column ranges.
hmmconvert   Convert profile formats to/from HMMER3 format.
hmmemit   Generate (sample) sequences from a profile HMM.
hmmfetch   Get a profile HMM by name or accession from an HMM database.
hmmpress   Format an HMM database into a binary format for `hmmscan`.
hmmstat   Show summary statistics for each profile in an HMM database.

The quadrumvirate of `hmmbuild/hmmsearch/hmmscan/hmmalign` is the core functionality for protein domain analysis and annotation pipelines, for instance using profile databases like Pfam or SMART.

The `phmmer` and `jackhmmer` programs search a single protein sequence against a protein sequence database, akin to BLASTP and PSIBLAST, respectively. (Internally, they just produce a profile HMM from the query sequence, then run HMM searches.)

The pair `nhmmer/nhmmscan` are new to HMMER3.1. The program `nhmmer` can search against a DNA sequence database with a query of a prebuilt HMM (built using `hmmbuild`), a multiple sequence alignment, or a single sequence. The program `nhmmscan` can search an HMM database with a DNA sequence query.

The program `hmmpgmd` is also new to HMMER3.1. It is the daemon that we use internally for the hmmer.org web server, and essentially stands in front of the protein search tools `phmmer`, `hmmsearch`, and `hmmscan`. As a daemon, it starts up, loads the target database into memory, then performs searches against that database as requested by client programs.

In the Tutorial section, we'll show examples of running each of these programs, using examples in the `tutorial/` subdirectory of the distribution.

Supported formats

HMMER can usually automatically detect the format of a multiple sequence alignment, for example given to `hmmbuild` or as the query to `nhmmer`. To override autodetection, specify `--informat afa` on the command line of any program that reads an input alignment.

HMMER can also usually detect the format of an unaligned sequence file, for example given as the target database to one of the search programs, or as input to `hmmalign`. Autodetection can be overridden with program-specifc flags, for example specifying `--tformat afa` on the command line of search programs.

See man-pages for program-specific lists of accepted formats.

Files used in the tutorial

The subdirectory `/tutorial` in the HMMER distribution contains the files used in the tutorial, as well as a number of examples of various file formats that HMMER reads. The important files for the tutorial are:

globins4.sto An example alignment of four globin sequences, in Stockholm format. This alignment is a subset of a famous old published structural alignment from Don Bashford (Bashford et al., 1987).

globins4.hmm An example profile HMM file, built from `globins4.sto`, in HMMER3 ASCII text format.

globins4.out An example `hmmsearch` output file that results from searching the `globins4.hmm` against UniProt 15.7.

globins45.fa A FASTA file containing 45 unaligned globin sequences.

fn3.sto An example alignment of 106 fibronectin type III domains. This is the Pfam 22.0 `fn3` seed alignment. It provides an example of a Stockholm format with more complex annotation. We'll also use it for an example of `hmmsearch` analyzing sequences containing multiple domains.

fn3.hmm A profile HMM created from `fn3.sto` by `hmmbuild`.

7LESS_DROME A FASTA file containing the sequence of the *Drosophila* Sevenless protein, a receptor tyrosine kinase whose extracellular region is thought to contain seven fibronectin type III domains.

fn3.out Output of `hmmsearch fn3.hmm 7LESS_DROME`.

Pkinase.sto The Pfam 22.0 Pkinase seed alignment of protein kinase domains.

Pkinase.hmm A profile HMM created from `Pkinase.sto` by `hmmbuild`.

minifam An example HMM flatfile database, containing three models: `globins4`, `fn3`, and `Pkinase`.

minifam.h3{m,i,f,p} Binary compressed files corresponding to `minifam`, produced by `hmmpress`.

HBB_HUMAN A FASTA file containing the sequence of human $\beta-$hemoglobin, used as an example query for `phmmer` and `jackhmmer`.

MADE1.sto An example alignment of 1997 instances of the MADE1 transposable element family. This is the Dfam 1.1 `MADE1` seed alignment. We'll use it for an example of using `nhmmer` to search a DNA database.

MADE1.hmm A profile HMM created from `MADE1.sto` by `hmmbuild` with default parameters (note: this model differs from the MADE1 model found in DFAM, which is built with more.

dna_target.fa A FASTA file containing 330000 bases extracted from human chromosome 1, in which four MADE1 instances are found. This is used as the example database for `nhmmer` and `nhmmscan`.

Searching a protein sequence database with a single protein profile HMM

Step 1: build a profile HMM with hmmbuild

HMMER starts with a multiple sequence alignment file that you provide. The format can usually be autodetected, with accepted formats described above. The file `tutorial/globins4.sto` is an example of a simple Stockholm file. It looks like this:

```
# STOCKHOLM 1.0

HBB_HUMAN     ........VHLTPEEKSAVTALWGKV....NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVL
HBA_HUMAN     .........VLSPADKTNVKAAWGKVGA..HAGEYGAEALERMFLSFPTTKTYFPHF.DLS.....HGSAQVKGHGKKVA
MYG_PHYCA     .........VLSEGEWQLVLHVWAKVEA..DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASEDLKKHGVTVL
GLB5_PETMA    PIVDTGSVAPLSAAEKTKIRSAWAPVYS..TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSADVRWHAERII

HBB_HUMAN     GAFSDGLAHL...D..NLKGTFATLSELHCDKL..HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANAL
HBA_HUMAN     DALTNAVAHV...D..DMPNALSALSDLHAHKL..RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVL
MYG_PHYCA     TALGAILKK....K.GHHEAELKPLAQSHATKH..KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDI
GLB5_PETMA    NAVNDAVASM..DDTEKMSMKLRDLSGKHAKSF..QVDPQYFKVLAAVIADTVAAG.........DAGFEKLMSMICILL

HBB_HUMAN     AHKYH......
HBA_HUMAN     TSKYR......
MYG_PHYCA     AAKYKELGYQG
GLB5_PETMA    RSAY.......
//
```

Most popular alignment formats are similar block-based formats, and can be turned into Stockholm format with a little editing or scripting. Don't forget the `# STOCKHOLM 1.0` line at the start of the alignment, nor the `//` at the end. Stockholm alignments can be concatenated to create an alignment database flatfile containing many alignments.

The `hmmbuild` command builds a profile HMM from an alignment (or HMMs for each of many alignments in a Stockholm file), and saves the HMM(s) in a file. For example, type:

> **hmmbuild globins4.hmm tutorial/globins4.sto**

and you'll see some output that looks like:

```
# hmmbuild :: profile HMM construction from multiple sequence alignments
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# input alignment file:         globins4.sto
# output HMM file:              globins4.hmm
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# idx name                 nseq  alen  mlen eff_nseq re/pos description
#---- -------------------- ----- ----- ----- -------- ------ -----------
1     globins4                 4   171   149     0.96  0.589

# CPU time: 0.24u 0.00s 00:00:00.24 Elapsed: 00:00:00.27
```

If your input file had contained more than one alignment, you'd get one line of output for each model. For instance, a single `hmmbuild` command suffices to turn a Pfam seed alignment flatfile (such as `Pfam-A.seed`) into a profile HMM flatfile (such as `Pfam.hmm`).

The information on these lines is almost self-explanatory. The `globins4` alignment consisted of 4 sequences with 171 aligned columns. HMMER turned it into a model of 149 consensus positions, which

means it defined 22 gap-containing alignment columns to be insertions relative to consensus. The 4 sequences were only counted as an "effective" total sequence number (`eff_nseq`) of 0.96. The model ended up with a relative entropy per position ((`re/pos`; information content) of 0.589 bits.

The new HMM was saved to `globins4.hmm`. If you were to look at this file (and you don't have to – it's intended for HMMER's consumption, not yours), you'd see something like:

```
HMMER3/f [3.1 | February 2013]
NAME  globins4
LENG  149
ALPH  amino
RF    no
MM    no
CONS  yes
CS    no
MAP   yes
DATE  Thu Feb 14 16:44:36 2013
NSEQ  4
EFFN  0.964844
CKSUM 2027839109
STATS LOCAL MSV      -9.9014  0.70957
STATS LOCAL VITERBI -10.7224  0.70957
STATS LOCAL FORWARD  -4.1637  0.70957
HMM         A        C        D        E        F        G        H        I        K        L        M        N        P
          m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO  2.36553  4.52577  2.96709  2.70473  3.20818  3.02239  3.41069  2.90041  2.55332  2.35210  3.67329  3.19812  3.45
         2.68640  4.42247  2.77497  2.73145  3.46376  2.40504  3.72516  3.29302  2.67763  2.69377  4.24712  2.90369  2.73
         0.57544  1.78073  1.31293  1.75577  0.18968  0.00000       *
      1  1.70038  4.17733  3.76164  3.36686  3.72281  3.29583  4.27570  2.40482  3.29230  2.54324  3.63799  3.55099  3.93
         2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354  2.67741  2.69355  4.24690  2.90347  2.73
         0.03156  3.86736  4.58970  0.61958  0.77255  0.34406  1.23405
...
    149  2.92198  5.11574  3.28049  2.65489  4.47826  3.59727  2.51142  3.88373  1.57593  3.35205  4.19259  3.10178  3.96
         2.68634  4.42241  2.77536  2.73098  3.46370  2.40469  3.72511  3.29370  2.67757  2.69331  4.24706  2.90363  2.73
         0.22163  1.61553        *  1.50361  0.25145  0.00000       *
//
```

The HMMER ASCII save file format is defined in Section 8.

Step 2: search the sequence database with hmmsearch

Presumably you have a sequence database to search. Here we'll use the UniProt 2013_02 Swiss-Prot FASTA format flatfile (not provided in the tutorial, because of its large size), `uniprot_sprot.fasta`. If you don't have a sequence database handy, run your example search against `tutorial/globins45.fa` instead, which is a FASTA format file containing 45 globin sequences.

`hmmsearch` accepts any FASTA file as target database input. It also accepts EMBL/UniProt text format, and Genbank format. It will automatically determine what format your file is in; you don't have to say. An example of searching a sequence database with our `globins4.hmm` model would look like:

> **hmmsearch globins4.hmm uniprot_sprot.fasta > globins4.out**

Depending on the database you search, the output file `globins4.out` should look more or less like the example of a UniProt search output provided in `tutorial/globins4.out`.

The first section is the *header* that tells you what program you ran, on what, and with what options:

```
# hmmsearch :: search profile(s) against a sequence database
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query HMM file:                 globins4.hmm
# target sequence database:       uniprot\_sprot.fasta
# per-seq hits tabular output:    globins4.tbl
# per-dom hits tabular output:    globins4.domtbl
# number of worker threads:       2
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       globins4  [M=149]
Scores for complete sequences (score includes all domains):
```

The second section is the *sequence top hits* list. It is a list of ranked top hits (sorted by E-value, most significant hit first), formatted in a BLAST-like style:

```
 --- full sequence ---    --- best 1 domain ---    -#dom-
 E-value  score  bias    E-value  score  bias    exp  N  Sequence               Description
 -------  ------ -----    -------  ------ -----    ---- --  --------               -----------
 6.5e-65  222.7   3.2     7.2e-65  222.6   3.2    1.0  1  sp|P02185|MYG_PHYMC    Myoglobin OS=Physeter macrocephalus GN
 3.3e-63  217.2   0.1     3.7e-63  217.0   0.1    1.0  1  sp|P02024|HBB_GORGO    Hemoglobin subunit beta OS=Gorilla gor
 4.9e-63  216.6   0.0     5.4e-63  216.5   0.0    1.0  1  sp|P68871|HBB_HUMAN    Hemoglobin subunit beta OS=Homo sapien
 4.9e-63  216.6   0.0     5.4e-63  216.5   0.0    1.0  1  sp|P68872|HBB_PANPA    Hemoglobin subunit beta OS=Pan paniscu
 4.9e-63  216.6   0.0     5.4e-63  216.5   0.0    1.0  1  sp|P68873|HBB_PANTR    Hemoglobin subunit beta OS=Pan troglod
   7e-63  216.1   3.0     7.7e-63  216.0   3.0    1.0  1  sp|P02177|MYG_ESCGI    Myoglobin OS=Eschrichtius gibbosus GN=
 ...
```

The last two columns, obviously, are the name of each target sequence and optional description.

The most important number here is the first one, the *sequence E-value*. This is the statistical significance of the match to this sequence: the number of hits we'd expect to score this highly in a database of this size if the database contained only nonhomologous random sequences. The lower the E-value, the more significant the hit.

The E-value is based on the *sequence bit score*, which is the second number. This is the log-odds score for the complete sequence. Some people like to see a bit score instead of an E-value, because the bit score doesn't depend on the size of the sequence database, only on the profile HMM and the target sequence.

The next number, the *bias*, is a correction term for biased sequence composition that has been applied to the sequence bit score.[1] For instance, for the top hit MYG_PHYCA that scored 222.7 bits, the bias of 3.2 bits means that this sequence originally scored 225.9 bits, which was adjusted by the slight 3.2 bit biased-composition correction. The only time you really need to pay attention to the bias value is when it's large, on the same order of magnitude as the sequence bit score. Sometimes (rarely) the bias correction isn't aggressive enough, and allows a non-homolog to retain too much score. Conversely, the bias correction can be too aggressive sometimes, causing you to miss homologs. You can turn the biased-composition score correction off with the `--nonull2` option (and if you're doing that, you may also want to set `--nobias`, to turn off another biased composition step called the bias filter, which affects which sequences get scored at all).

The next three numbers are again an E-value, score, and bias, but only for the single best-scoring domain in the sequence, rather than the sum of all its identified domains. The rationale for this isn't apparent in the globin example, because all the globins in this example consist of only a single globin domain. So let's set up a second example, using a model of a single domain that's commonly found in multiple domains in a single sequence. Build a fibronectin type III domain model using the `tutorial/fn3.sto` alignment (this happens to be a Pfam seed alignment; it's a good example of an alignment with complex Stockholm annotation). Then use that model to analyze the sequence `tutorial/7LESS_DROME`, the *Drosophila* Sevenless receptor tyrosine kinase:

> **hmmbuild fn3.hmm tutorial/fn3.sto**

> **hmmsearch fn3.hmm tutorial/7LESS_DROME > fn3.out**

An example of what that output file will look like is provided in `tutorial/fn3.out`. The sequence top hits list says:

```
 --- full sequence ---    --- best 1 domain ---    -#dom-
 E-value  score  bias    E-value  score  bias    exp  N  Sequence      Description
 -------  ------ -----    -------  ------ -----    ---- --  --------      -----------
 1.9e-57  178.0   0.4     1.2e-16   47.2   0.9    9.4  9  7LESS_DROME   RecName: Full=Protein sevenless;
```

OK, now let's pick up the explanation where we left off. The total sequence score of 178.0 sums up *all* the fibronectin III domains that were found in the 7LESS_DROME sequence. The "single best dom" score and E-value are the bit score and E-value as if the target sequence only contained the single best-scoring domain, without this summation.

The idea is that we might be able to detect that a sequence is a member of a multidomain family because it contains multiple weakly-scoring domains, even if no single domain is solidly significant on its own. On

---

[1]The method that HMMER uses to compensate for biased composition is unpublished. We will write it up when there's a chance.

the other hand, if the target sequence happened to be a piece of junk consisting of a set of identical internal repeats, and one of those repeats accidentally gives a weak hit to the query model, all the repeats will sum up and the sequence score might look "significant" (which mathematically, alas, is the correct answer: the null hypothesis we're testing against is that the sequence is a *random* sequence of some base composition, and a repetitive sequence isn't random).

So operationally:

- if both E-values are significant ($<< 1$), the sequence is likely to be homologous to your query.

- if the full sequence E-value is significant but the single best domain E-value is not, the target sequence is probably a multidomain remote homolog; but be wary, and watch out for the case where it's just a repetitive sequence.

OK, the sharp eyed reader asks, if that's so, then why in the globin4 output (all of which have only a single domain) do the full sequence bit scores and best single domain bit scores not exactly agree? For example, the top ranked hit, MYG_PHYCA (sperm whale myoglobin, if you're curious) has a full sequence score of 222.7 and a single best domain score of 222.6. What's going on? What's going on is that the position and alignment of that domain is uncertain – in this case, only very slightly so, but nonetheless uncertain. The full sequence score is summed over all possible alignments of the globin model to the MYG_PHYCA sequence. When HMMER identifies domains, it identifies what it calls an **envelope** bounding where the domain's alignment most probably lies. (More on this later, when we discuss the reported coordinates of domains and alignments in the next section of the output.) The "single best dom" score is calculated after the domain envelope has been defined, and the summation is restricted only to the ensemble of possible alignments that lie within the envelope. The fact that the two scores are slightly different is therefore telling you that there's a small amount of probability (uncertainty) that the domain lies somewhat outside the envelope bounds that HMMER has selected.

The two columns headed #doms are two different estimates of the number of distinct domains that the target sequence contains. The first, the column marked exp, is the *expected* number of domains according to HMMER's statistical model. It's an average, calculated as a weighted marginal sum over all possible alignments. Because it's an average, it isn't necessarily a round integer. The second, the column marked N, is the number of domains that HMMER's domain postprocessing and annotation pipeline finally decided to identify, annotate, and align in the target sequence. This is the number of alignments that will show up in the domain report later in the output file.

These two numbers should be about the same. Rarely, you might see that they're wildly different, and this would usually be a sign that the target sequence is so highly repetitive that it's confused the HMMER domain postprocessors. Such sequences aren't likely to show up as significant homologs to any sensible query in the first place.

The sequence top hits output continues until it runs out of sequences to report. By default, the report includes all sequences with an E-value of 10.0 or less.

Then comes the third output section, which starts with

```
Domain annotation for each sequence (and alignments):
```

Now for each sequence in the top hits list, there will be a section containing a table of where HMMER thinks all the domains are, followed by the alignment inferred for each domain. Let's use the fn3 vs. 7LESS_DROME example, because it contains lots of domains, and is more interesting in this respect than the

globin4 output. The domain table for `7LESS_DROME` looks like:

```
>> 7LESS_DROME  RecName: Full=Protein sevenless;       EC=2.7.10.1;
   #    score  bias  c-Evalue  i-Evalue hmmfrom  hmm to    alifrom  ali to    envfrom  env to    acc
 ---   ------ ----- --------- --------- -------  -------   -------  -------   -------  -------   ----
   1 ?   -1.3   0.0      0.17      0.17      61      74 ..     396      409 ..     395      411 .. 0.85
   2 !   40.7   0.0   1.3e-12   1.3e-14       2      84 ..     439      520 ..     437      521 .. 0.95
   3 !   14.4   0.0     2e-06     2e-06      13      85 ..     836      913 ..     826      914 .. 0.73
   4 !    5.1   0.0    0.0016    0.0016      10      36 ..    1209     1235 ..    1203     1259 .. 0.82
   5 !   24.3   0.0   1.7e-09   1.7e-09      14      80 ..    1313     1380 ..    1304     1386 .. 0.82
   6 ?    0.0   0.0     0.063     0.063      58      72 ..    1754     1768 ..    1739     1769 .. 0.89
   7 !   47.2   0.9   1.2e-16   1.2e-16       1      85 [.    1799     1890 ..    1799     1891 .. 0.91
   8 !   17.8   0.0   1.8e-07   1.8e-07       6      74 ..    1904     1966 ..    1901     1976 .. 0.90
   9 !   12.8   0.0   6.6e-06   6.6e-06       1      86 []    1993     2107 ..    1993     2107 .. 0.89
```

Domains are reported in the order they appear in the sequence, not in order of their significance.

The `!` or `?` symbol indicates whether this domain does or does not satisfy both per-sequence and per-domain inclusion thresholds. Inclusion thresholds are used to determine what matches should be considered to be "true", as opposed to reporting thresholds that determine what matches will be reported (often including the top of the noise, so you can see what interesting sequences might be getting tickled by your search). By default, inclusion thresholds usually require a per-sequence E value of 0.01 or less and a per-domain conditional E-value of 0.01 or less (except jackhmmer, which requires a more stringent 0.001 for both), and reporting E-value thresholds are set to 10.0.

The bit score and bias values are as described above for sequence scores, but are the score of just one domain's envelope.

The first of the two E-values is the **conditional E-value**. This is an odd number, and it's not even clear we're going to keep it. Pay attention to what it means! It is an attempt to measure the statistical significance of each domain, *given that we've already decided that the target sequence is a true homolog*. It is the expected number of *additional* domains we'd find with a domain score this big in the set of sequences reported in the top hits list, if those sequences consisted only of random nonhomologous sequence outside the region that sufficed to define them as homologs.

The second number is the **independent E-value**: the significance of the sequence in the *whole* database search, if this were the only domain we had identified. It's exactly the same as the "best 1 domain" E-value in the sequence top hits list.

The different between the two E-values is not apparent in the `7LESS_DROME` example because in both cases, the size of the search space as 1 sequence. There's a single sequence in the target sequence database (that's the size of the search space that the independent/best single domain E-value depends on). There's one sequence reported as a putative homolog in the sequence top hits list (that's the size of the search space that the conditional E-value depends on). A better example is to see what happens when we search UniProt (2013_02 contains 539165 sequences) with the `fn3` model:

> **`hmmsearch fn3.hmm uniprot_sprot.fasta`**

(If you don't have UniProt and can't run a command like this, don't worry about it - I'll show the relevant bits here.) Now the domain report for `7LESS_DROME` looks like:

```
   #    score  bias  c-Evalue  i-Evalue hmmfrom  hmm to    alifrom  ali to    envfrom  env to    acc
 ---   ------ ----- --------- --------- -------  -------   -------  -------   -------  -------   ----
   1 !   40.7   0.0   9.6e-12   6.9e-09       2      84 ..     439      520 ..     437      521 .. 0.95
   2 !   14.4   0.0    0.0015       1.1      13      85 ..     836      913 ..     826      914 .. 0.73
   3 ?    5.1   0.0       1.2   8.6e+02      10      36 ..    1209     1235 ..    1203     1259 .. 0.82
   4 !   24.3   0.0   1.3e-06   0.00091      14      80 ..    1313     1380 ..    1304     1386 .. 0.82
   5 !   47.2   0.9   8.8e-14   6.3e-11       1      85 [.    1799     1890 ..    1799     1891 .. 0.91
   6 !   17.8   0.0   0.00014     0.099       6      74 ..    1904     1966 ..    1901     1976 .. 0.90
   7 !   12.8   0.0     0.005       3.5       1      86 []    1993     2107 ..    1993     2107 .. 0.89
```

Notice that *almost* everything's the same (it's the same target sequence, after all) *except* for what happens with E-values. The independent E-value is calculated assuming a search space of all 539165 sequences. For example, look at the highest scoring domain (domain 5 here; domain 7 above). When we only looked at a single sequence, its score of 47.2 bits has an E-value of 1.2e-16. When we search a database of 539165 sequences, a hit scoring 47.2 bits would be expected to happen 539165 times as often: $1.2\text{e-}16 \times 539165 = 6.47\text{e-}11$ (it's showing 6.3e-11 because of roundoff issues; the 1.2e-16 in fact isn't

exactly 1.2e-16 inside HMMER). In this UniProt search, 754 sequences were reported in the top hits list (with E-values ≤ 10). If we were to assume that all 754 are true homologs, x out the domain(s) that made us think that, and then went looking for *additional* domains in those 754 sequences, we'd be searching a smaller database of 754 sequences: the expected number of times we'd see a hit of 47.2 bits or better is now 1.2e-16 $\times$ 754 = 9.0e-14. That's where the conditional E-value (c-Evalue) is coming from (subject to rounding error).

Notice that a couple of domains disappeared in the UniProt search, because now, in this larger search space size, they're not significant. Domain 1 (the one with the score of -1.3 bits) got a conditional E-value of 0.17 $\times$ 754 = 128, and domain 6 (with a bit score of 0.0) got a c-Evalue of 0.063 $\times$ 754 = 47.5. These fail the default reporting threshold of 10.0. The domain with a score of 5.1 bits also shifted from being above to below the default inclusion thresholds, so now it's marked with a ? instead of a !.

Operationally:

- If the independent E-value is significant ($<< 1$), that means that even this single domain *by itself* is such a strong hit that it suffices to identify the sequence as a significant homolog with respect to the size of the entire original database search. You can be confident that this is a homologous domain.

- Once there's one or more high-scoring domains in the sequence already, sufficient to decide that the sequence contains homologs of your query, you can look (with some caution) at the conditional E-value to decide the statistical significance of additional weak-scoring domains.

In the UniProt output, for example, we'd be pretty sure of four of the domains (1, 4, 5, and maybe 6), each of which has a strong enough independent E-value to declare 7LESS_DROME to be an fnIII-domain-containing protein. Domains 2 and 7 wouldn't be significant if they were all we saw in the sequence, but once we decide that 7LESS_DROME contains fn3 domains on the basis of the other hits, their conditional E-values indicate that they are probably also fn3 domains too. Domain 3 is too weak to be sure of, from this search alone, but would be something to pay attention to.

The next four columns give the endpoints of the reported local alignment with respect to both the query model ("hmm from" and "hmm to") and the target sequence ("ali from" and "ali to").

It's not easy to tell from the "to" coordinate whether the alignment ended internally in the query or target, versus ran all the way (as in a full-length global alignment) to the end(s). To make this more readily apparent, with each pair of query and target endpoint coordinates, there's also a little symbology. .. meaning both ends of the alignment ended internally, and [] means both ends of the alignment were full-length flush to the ends of the query or target, and [. and .] mean only the left or right end was flush/full length.

The next two columns ("env from" and "env to") define the *envelope* of the domain's location on the target sequence. The envelope is almost always a little wider than what HMMER chooses to show as a reasonably confident alignment. As mentioned earlier, the envelope represents a subsequence that encompasses most of the posterior probability for a given homologous domain, even if precise endpoints are only fuzzily inferrable. You'll notice that for higher scoring domains, the coordinates of the envelope and the inferred alignment will tend to be in tighter agreement, corresponding to sharper posterior probability defining the location of the homologous region.

Operationally, we would use the envelope coordinates to annotate domain locations on target sequences, not the alignment coordinates. However, be aware that when two weaker-scoring domains are close to each other, envelope coordinates can and will overlap, corresponding to the overlapping uncertainty of where one domain ends and another begins. In contrast, alignment coordinates generally do not overlap (though there are cases where even they will overlap[2]). In contrast, alignment coordinates generally do not overlap (though there are cases where even they will overlap[2]).

The last column is the average posterior probability of the aligned target sequence residues; effectively, the expected accuracy per residue of the alignment.

---

[2]Not to mention one (mercifully rare) bug/artifact that we're betting is so unusual that testers don't even see an example of it – but we'll see.

For comparison, current UniProt consensus annotation of Sevenless shows seven domains:

```
FT   DOMAIN      311    431      Fibronectin type-III 1.
FT   DOMAIN      436    528      Fibronectin type-III 2.
FT   DOMAIN      822    921      Fibronectin type-III 3.
FT   DOMAIN     1298   1392      Fibronectin type-III 4.
FT   DOMAIN     1680   1794      Fibronectin type-III 5.
FT   DOMAIN     1797   1897      Fibronectin type-III 6.
FT   DOMAIN     1898   1988      Fibronectin type-III 7.
```

These domains are a pretty tough case to call, actually. HMMER fails to see anything significant overlapping two of these domains (311-431 and 1680-1794) in the UniProt search, though it sees a smidgen of them when 7LESS_DROME alone is the target. HMMER3 sees two new domains (1205-1235 and 1993-2098) that UniProt currently doesn't annotate, but these are pretty plausible domains (given that the extracellular domain of Sevenless is pretty much just a big array of ∼100aa fibronectin repeats.

Under the domain table, an "optimal posterior accuracy" alignment (Holmes, 1998) is computed within each domain's envelope, and displayed. For example, (skipping domain 1 because it's weak and unconvincing), fibronectin III domain 2 in your 7LESS_DROME output is shown as:

```
== domain 2    score: 40.7 bits;  conditional E-value: 1.3e-14
               ---CEEEEEEECTTEEEEEEE--S..SS--SEEEEEEEETTTCCGCEEEEEETTTSEEEEES--TT-EEEEEEEEEETTEE.E CS
        fn3  2 saPenlsvsevtstsltlsWsppkdgggpitgYeveyqekgegeewqevtvprtttsvtltgLepgteYefrVqavngagegp 84
               saP  ++ +  ++ l ++W p +  +gpi+gY++++++++++  + e+ vp+   s+ +++L++gt+Y++ +  +n++gegp
7LESS_DROME 439 SAPVIEHLMGLDDSHLAVHWHPGRFTNGPIEGYRLRLSSSEGNA-TSEQLVPAGRGSYIFSQLQAGTNYTLALSMINKQGEGP 520
               78999999999******************************9998.********************************9997 PP
```

The initial header line starts with a == as a little handle for a parsing script to grab hold of. We may put more information on that line eventually.

If the model had any consensus structure or reference line annotation that it inherited from your multiple alignment (#=GC SS_cons, #=GC RF annotation in Stockholm files), that information is simply regurgitated as CS or RF annotation lines here. The fn3 model had a consensus structure annotation line.

The line starting with fn3 is the consensus of the query model. Capital letters represent the most conserved (high information content) positions. Dots (.) in this line indicate insertions in the target sequence with respect to the model.

The midline indicates matches between the query model and target sequence. A + indicates positive score, which can be interpreted as "conservative substitution", with respect to what the model expects at that position.

The line starting with 7LESS_DROME is the target sequence. Dashes (-) in this line indicate deletions in the target sequence with respect to the model.

The bottom line represents the posterior probability (essentially the expected accuracy) of each aligned residue. A 0 means 0-5%, 1 means 5-15%, and so on; 9 means 85-95%, and a * means 95-100% posterior probability. You can use these posterior probabilities to decide which parts of the alignment are well-determined or not. You'll often observe, for example, that expected alignment accuracy degrades around locations of insertion and deletion, which you'd intuitively expect.

You'll also see expected alignment accuracy degrade at the ends of an alignment – this is because "alignment accuracy" posterior probabilities currently not only includes whether the residue is aligned to one model position versus others, but also confounded with whether a residue should be considered to be homologous (aligned to the model somewhere) versus not homologous at all.[3]

These domain table and per-domain alignment reports for each sequence then continue, for each sequence that was in the per-sequence top hits list.

Finally, at the bottom of the file, you'll see some summary statistics. For example, at the bottom of the

---

[3]It may make more sense to condition the posterior probabilities on the assumption that the residue is indeed homologous: given that, how likely is it that we've got it correctly aligned.

globins search output, you'll find something like:

```
        Internal pipeline statistics summary:
        -------------------------------------
        Query model(s):                             1  (149 nodes)
        Target sequences:                      539165  (191456931 residues searched)
        Passed MSV filter:                      20801  (0.03858); expected 10783.3 (0.02)
        Passed bias filter:                     17061  (0.0316434); expected 10783.3 (0.02)
        Passed Vit filter:                       2321  (0.0043048); expected 539.2 (0.001)
        Passed Fwd filter:                       1109  (0.00205688); expected 5.4 (1e-05)
        Initial search space (Z):              539165  [actual number of targets]
        Domain search space  (domZ):             1108  [number of targets reported over threshold]
        # CPU time: 6.50u 0.11s 00:00:06.61 Elapsed: 00:00:02.59
        # Mc/sec: 11014.32
        //
```

This gives you some idea of what's going on in HMMER's acceleration pipeline. You've got one query HMM, and the database has 539,165 target sequences. Each sequence goes through a gauntlet of three scoring algorithms called MSV, Viterbi, and Forward, in order of increasing sensitivity and increasing computational requirement.

MSV (the "Multi ungapped Segment Viterbi" algorithm) essentially calculates the HMM equivalent of BLAST's sum score – an optimal sum of ungapped high-scoring alignment segments. Unlike BLAST, it does this calculation directly, without BLAST's word hit or hit extension step, using a SIMD vector-parallel algorithm. By default, HMMER is configured to allow sequences with a P-value of ≤ 0.02 through the MSV score filter (thus, if the database contained no homologs and P-values were accurately calculated, the highest scoring 2% of the sequences will pass the filter). Here, about 4% of the database got through the MSV filter.

A quick check is then done to see if the target sequence is "obviously" so biased in its composition that it's unlikely to be a true homolog. This is called the "bias filter". If you don't like it (it can occasionally be overaggressive) you can shut it off with the `--nobias` option. Here, 17061 sequences pass through the bias filter.

The Viterbi filter then calculates a gapped optimal alignment score. This is a bit more sensitive than the MSV score, but the Viterbi filter is about four-fold slower than MSV. By default, HMMER3 lets sequences with a P-value of ≤ 0.001 through this stage. Here (because there's a little over a thousand true globin homologs in this database), much more than that gets through - 2321 sequences.

Then the full Forward score is calculated, which sums over all possible alignments of the profile to the target sequence. The default allows sequences with a P-value of $\leq 10^{-5}$ through; 1109 sequences passed.

All sequences that make it through the three filters are then subjected to a full probabilistic analysis using the HMM Forward/Backward algorithms, first to identify domains and assign domain envelopes; then within each individual domain envelope, Forward/Backward calculations are done to determine posterior probabilities for each aligned residue, followed by optimal accuracy alignment. The results of this step are what you finally see on the output.

Recall the difference between conditional and independent E-values, with their two different search space sizes. These search space sizes are reported in the statistics summary.

Finally, it reports the speed of the search in units of Mc/sec (million dynamic programming cells per second), the CPU time, and the elapsed time. This search took about 2.59 seconds of elapsed (wall clock time) (running with `--cpu 2` on two cores). That's in the same ballpark as BLAST. On the same machine, also running dual-core, NCBI BLAST with one of these globin sequences took 2.3 seconds, and WU-BLAST took 4.8 seconds.

Single sequence protein queries using phmmer

The `phmmer` program is for searching a single sequence query against a sequence database, much as `BLASTP` or `FASTA` would do. `phmmer` works essentially just like `hmmsearch` does, except you provide a query sequence instead of a query profile HMM.

Internally, HMMER builds a profile HMM from your single query sequence, using a simple position-independent scoring system (BLOSUM62 scores converted to probabilities, plus a gap-open and gap-extend probability).

The file `tutorial/HBB_HUMAN` is a FASTA file containing the human $\beta-$globin sequence as an example query. If you have a sequence database such as `uniprot_sprot.fasta`, make that your target database; otherwise, use `tutorial/globins45.fa` as a small example:

> **phmmer tutorial/HBB_HUMAN uniprot_sprot.fasta**

or

> **phmmer tutorial/HBB_HUMAN tutorial/globins45.fa**

Everything about the output is essentially as previously described for `hmmsearch`.

Iterative protein searches using jackhmmer

The `jackhmmer` program is for searching a single sequence query iteratively against a sequence database, much as `PSI-BLAST` would do.

The first round is identical to a `phmmer` search. All the matches that pass the inclusion thresholds are put in a multiple alignment. In the second (and subsequent) rounds, a profile is made from these results, and the database is searched again with the profile.

Iterations continue either until no new sequences are detected or the maximum number of iterations is reached. By default, the maximum number of iterations is 5; you can change this with the `-N` option.

Your original query sequence is always included in the multiple alignments, whether or not it appears in the database.[4] The "consensus" columns assigned to each multiple alignment always correspond exactly to the residues of your query, so the coordinate system of every profile is always the same as the numbering of residues in your query sequence, 1..L for a sequence of length L.

Assuming you have UniProt or something like it handy, here's an example command line for a jackhmmer search:

> **jackhmmer tutorial/HBB_HUMAN uniprot_sprot.fasta**

One difference from `phmmer` output you'll notice is that `jackhmmer` marks "new" sequences with a $+$ and "lost" sequences with a $-$. New sequences are sequences that pass the inclusion threshold(s) in this round, but didn't in the round before. Lost sequences are the opposite: sequences that passed the inclusion threshold(s) in the previous round, but have now fallen beneath (yet are still in the reported hits – it's possible, though rare, to lose sequences utterly, if they no longer even pass the reporting threshold(s)). In the first round, everything above the inclusion thresholds is marked with a $+$, and nothing is marked with

---

[4]If it *is* in the database, it will almost certainly be included in the internal multiple alignment twice, once because it's the query and once because it's a significant database match to itself. This redundancy won't screw up the alignment, because sequences are downweighted for redundancy anyway.

a `-`. For example, the top of this output looks like:

```
# jackhmmer :: iteratively search a protein sequence against a protein database
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query sequence file:              HBB_HUMAN
# target sequence database:         uniprot_sprot.fasta
# per-seq hits tabular output:      hbb-jack.tbl
# per-dom hits tabular output:      hbb-jack.domtbl
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       HBB_HUMAN  [L=146]
Description: Human beta hemoglobin.

Scores for complete sequences (score includes all domains):
   --- full sequence ---   --- best 1 domain ---    -#dom-
    E-value  score  bias    E-value  score  bias    exp  N  Sequence              Description
    -------  -----  -----   -------  -----  -----   ---- --  --------              -----------
 +  3.3e-98  330.5  0.6     3.7e-98  330.3  0.6     1.0  1  sp|P68871|HBB_HUMAN   Hemoglobin subunit beta OS=Homo sapien
 +  3.3e-98  330.5  0.6     3.7e-98  330.3  0.6     1.0  1  sp|P68872|HBB_PANPA   Hemoglobin subunit beta OS=Pan paniscu
 +  3.3e-98  330.5  0.6     3.7e-98  330.3  0.6     1.0  1  sp|P68873|HBB_PANTR   Hemoglobin subunit beta OS=Pan troglod
 +  9.5e-98  329.0  0.7     1.1e-97  328.8  0.7     1.0  1  sp|P02024|HBB_GORGO   Hemoglobin subunit beta OS=Gorilla gor
 +  2.9e-96  324.2  0.5     3.2e-96  324.0  0.5     1.0  1  sp|P02025|HBB_HYLLA   Hemoglobin subunit beta OS=Hylobates l
 +  2.9e-95  320.9  0.6     3.2e-95  320.8  0.6     1.0  1  sp|P02032|HBB_SEMEN   Hemoglobin subunit beta OS=Semnopithec
...
```

That continues until the inclusion threshold is reached, at which point you see a tagline "inclusion threshold" indicating where the threshold was set:

```
 +  0.00047  25.0   0.2    0.00055  24.8   0.2    1.0  1  sp|Q0KIY5|MYG_KOGBR   Myoglobin OS=Kogia breviceps GN=MB PE=
 +  0.0006   24.6   0.0    0.00071  24.4   0.0    1.0  1  sp|P14399|MYG_MUSAN   Myoglobin OS=Mustelus antarcticus GN=m
  ------ inclusion threshold ------
    0.001    23.9   0.3    0.011    20.5   0.3    2.0  1  sp|P81044|HBAZ_MACEU  Hemoglobin subunit zeta (Fragments) OS
    0.0013   23.5   0.0    0.0017   23.2   0.0    1.1  1  sp|O80405|LGB3_PEA    Leghemoglobin Lb120-1 OS=Pisum sativum
```

The domain output and search statistics are then shown just as in `phmmer`. At the end of this first iteration, you'll see some output that starts with `@@` (this is a simple tag that lets you search through the file to find the end of one iteration and the beginning of another):

```
@@ New targets included:   935
@@ New alignment includes: 936 subseqs (was 1), including original query
@@ Continuing to next round.

@@
@@ Round:                  2
@@ Included in MSA:        936 subsequences (query + 935 subseqs from 935 targets)
@@ Model size:             146 positions
@@
```

This (obviously) is telling you that the new alignment contains 936 sequences, your query plus 935 significant matches. For round two, it's built a new model from this alignment. Now for round two, it fires off what's essentially an `hmmsearch` of the target database with this new model:

```
Scores for complete sequences (score includes all domains):
   --- full sequence ---   --- best 1 domain ---    -#dom-
    E-value  score  bias    E-value  score  bias    exp  N  Sequence              Description
    -------  -----  -----   -------  -----  -----   ---- --  --------              -----------
    7.5e-68  232.1  0.2     8.3e-68  232.0  0.2     1.0  1  sp|P02055|HBB_MELME   Hemoglobin subunit beta OS=Meles meles
    1.1e-67  231.5  0.4     1.3e-67  231.4  0.4     1.0  1  sp|P81042|HBE_MACEU   Hemoglobin subunit epsilon OS=Macropus
    1.3e-67  231.3  0.3     1.5e-67  231.1  0.3     1.0  1  sp|P15449|HBB_MELCA   Hemoglobin subunit beta OS=Mellivora c
    1.9e-67  230.8  0.2     2.1e-67  230.6  0.2     1.0  1  sp|P68046|HBB_ODORO   Hemoglobin subunit beta OS=Odobenus ro
...
```

If you skim down through this output, you'll start seeing newly included sequences marked with +'s, such

as:

```
  ...
    9.4e-30  108.5   0.0      1e-29  108.4   0.0   1.0  1 sp|Q9DEP0|MYG_CRYAN   Myoglobin OS=Cryodraco antarcticus GN=
  +  1.4e-29  107.9   0.2    1.6e-29  107.8   0.2   1.0  1 sp|P14397|MYG_GALGA   Myoglobin OS=Galeorhinus galeus GN=mb
     2.4e-29  107.2   0.0    2.7e-29  107.0   0.0   1.0  1 sp|P02022|HBAM_LITCT  Hemoglobin heart muscle subunit alpha-
  +  4.8e-29  106.2   0.1    5.3e-29  106.1   0.1   1.0  1 sp|P14398|MYG_GALJA   Myoglobin OS=Galeorhinus japonicus GN=
     1.9e-28  104.3   0.0    2.3e-28  104.0   0.0   1.0  1 sp|P09106|HBAT_PAPAN  Hemoglobin subunit theta-1 OS=Papio an
     3.7e-28  103.4   0.3    4.8e-28  103.0   0.3   1.0  1 sp|P80017|GLBD_CAUAR  Globin D, coelomic OS=Caudina arenicol
     4.1e-28  103.2   0.0    5.1e-28  102.9   0.0   1.0  1 sp|P0C227|GLB_NERAL   Globin OS=Nerita albicilla PE=1 SV=1
     3.1e-25   93.8   0.2    3.4e-25   93.7   0.2   1.0  1 sp|P18979|HBA1_UROHA  Hemoglobin subunit alpha-1 (Fragment)
  +  4.2e-24   90.2   0.0      5e-24   89.9   0.0   1.0  1 sp|Q90W04|NGB_TETNG   Neuroglobin OS=Tetraodon nigroviridis
     8.4e-24   89.2   0.0      1e-23   89.0   0.0   1.0  1 sp|P59742|NGB1_ONCMY  Neuroglobin-1 OS=Oncorhynchus mykiss G
  +    1e-23   88.9   0.0    1.3e-23   88.7   0.0   1.0  1 sp|P59743|NGB2_ONCMY  Neuroglobin-2 OS=Oncorhynchus mykiss G
  ...
```

It's unusual to see sequences get lost (and marked with -), but it can happen; it doesn't happen in this globin example.

After round 2, many more globin sequences have been found:

```
@@ New targets included:   172
@@ New alignment includes: 1110 subseqs (was 936), including original query
@@ Continuing to next round.

@@
@@ Round:                  3
@@ Included in MSA:        1110 subsequences (query + 1109 subseqs from 1107 targets)
@@ Model size:             146 positions
@@
```

Because new sequences were included, it keeps going to round three, and then again to round four, then again to round five. After round five, the search ends quietly because there's a default maximum of five iterations, and you get:

```
@@ New targets included:   1
@@ New alignment includes: 1151 subseqs (was 1149), including original query
//
```

In this example, round 5 results in an alignment with two new subsequences, a new hit (sp—Q09240—GLOB9_CAEEL), and a second matching domain in a previously found hit (sp—P81044—HBAZ_MACEU). The final alignment includes sequences from 1150 hits, with one hit (HBAZ_MACEU) contributing two matching domains.

That // marks the end of the results for one query.

Searching a DNA sequence database

Step 1: Optionally build a profile HMM with hmmbuild

This step is nearly identical to Step 1 for protein profile HMM. For the DNA example, type:

> **hmmbuild MADE1.hmm tutorial/MADE1.sto**

and you'll see some output that looks like:

```
# hmmbuild :: profile HMM construction from multiple sequence alignments
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# input alignment file:            tutorial/MADE1.sto
# output HMM file:                 MADE1.hmm
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# idx name                 nseq  alen  mlen     W  eff_nseq re/pos description
#---- -------------------- ----- ----- ----- ----- -------- ------ -----------
1     MADE1                1997  1112    80   426      3.91  0.708 MADE1 (MAriner Derived Element 1), a TcMar-Mariner DNA

# CPU time: 0.15u 0.01s 00:00:00.16 Elapsed: 00:00:00.20
```

Notice the new output column with the header "W", which is only present when the input sequence alignment is made up of DNA or RNA. This represents an upper bound on the length at which nhmmer

expects to find an instance of the family[5]. It is always larger than mlen, though the ratio of mlen to W depends on the observed insert rate in the seed alignment. This length is used deep in the acceleration pipeline, and modest changes are not expected to impact results, but larger values of W do lead to longer run time. The value can be overridden with the `--w_length` or `--w_beta` flags, at the risk of possibly missing instances of the family that happen to be longer than W due to plentiful insertions.

Step 2: search the DNA sequence database with nhmmer

We'll use `tutorial/dna_target.fa` as the target sequence database. It is a FASTA format file containing one 330Kb long DNA sequence extracted from human chromosome 1.

The program `nhmmer` accepts a target DNA sequence database in the same formats as hmmsearch (we typically use FASTA). For the query, it accepts either an HMM file as produced above by hmmbuild, or a file containing either one DNA sequence or an alignment of multiple DNA sequences.

If a sequence or alignment is used as query input, `nhmmer` internally produces the HMM for that alignment[6], then searches with that model. The HMM produced in this way is automatically saved to disk; the default file name is chosen by appending ".hmm" to the name of the sequence file name. This can be overridden with the `--hmmout` flag.

An example of searching a sequence database with our `MADE1.hmm` model would look like:

> **nhmmer MADE1.hmm tutorial/dna_target.fa > MADE1.out**

The output file `MADE1.out` should look like the example provided in `tutorial/MADE1.out`.

This output is largely similar to that of hmmsearch. The key differences are that (1) each hit is not to a full sequence in the target database, but a local alignment of the HMM to a subsequence of a full target database sequence, and (2) there are no domains.

The first section is the *header* that tells you what program you ran, on what, and with what options, as above.

The second section is the *top hits* list. It is a list of ranked top hits (sorted by E-value, most significant hit first), formatted much like the `hmmsearch` output *top hits* list:

```
E-value  score  bias  Sequence                        start    end  Description
-------  ------ -----  --------                        -----  -----  -----------
8.4e-11   39.0   7.4  humanchr1/239220001-239550000  302390 302466
7.8e-08   29.5   6.0  humanchr1/239220001-239550000  302466 302390
8.4e-08   29.4   8.3  humanchr1/239220001-239550000  174456 174498
5.6e-06   23.6   7.0  humanchr1/239220001-239550000  174493 174456
------ inclusion threshold ------
          1.7    6.0   6.7  humanchr1/239220001-239550000  304074 304104
```

The table presents the *hit E-value*, *sequence bit score*, *bias*, *Sequence* and *Description*. See the section above for `hmmsearch` for a description of these fields.

The "start" and "end" columns give, for each hit, the range in the target sequence at which the hit is found. Note that "end" can be smaller than "start", indicating a hit found on the reverse complement of the target database sequence.

Note that one of the five reported hits falls below the inclusion threshold.

The observant reader will notice that the first two hits cover the same range of positions, one on the forward strand (302390..302466), the other on the reverse (302466..302390). The next two hits likewise cover a shared range. This happens because the MADE1 model is palindromic (the consensus is almost perfectly so), and highlights the important facts that (a) `nhmmer` searches on both strands of an input sequence, and (b) this can sometimes lead to overlapping opposite-strand hits, which are not filtered.

Then comes the third output section, which starts with

```
Annotation for each hit  (and alignments):
```

---

[5]W is based on position-specific insert rates: only $1e - 7$ of all sequences generated from the profile HMM will have length greater than W.

[6]Using default hmmbuild parameters; if you want more control, explicitly built the model with hmmbuild.

For each hit in the top hits list, there will be a one-line table providing detailed information about the hit, followed by the alignment inferred for the hit. The first entry from the MADE1 example above looks like:

```
>> humanchr1/239220001-239550000
    score  bias   Evalue   hmmfrom   hmm to    alifrom    ali to     envfrom   env to    sq len      acc
    ------ ----- --------- -------   -------   --------- ---------   --------- --------- ---------    ----
  !  39.0   7.4   8.4e-11        4        80 .]    302390    302466 ..    302387    302466 ..    330000    0.87
```

The bit score, bias value, Evalue, and acc are as described for hmmsearch, as is the choice of ! or ? symbols.

The next four columns give the endpoints of the reported local alignment with respect to both the query model ("hmm from" and "hmm to") and the target sequence ("ali from" and "ali to"). These are as described in the section for hmmsearch results, including the symbology used to recognize flush vs internal end points in hits.

The next two columns ("env from" and "env to") also behave as described earlier for hmmsearch, defining the *envelope* of the hit's location on the target sequence.

The "sq len" column indicates the full length of the target sequence, enabling simple calculation of the proximity of a hit to the end of the target.

Under each one-line hit table is displayed the alignment inferred between the model and the hit envelope. For example, the top hit from above is shown as:

```
    Alignment:
    score: 39.0 bits
                                  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx....xxxxxxxxxxxxxxxxxxxxxxxxxxx RF
                  MADE1      4 ggttggtgcaaaagtaattgcggtttttgccattacttttaatggc....aaaaaccgcaattacttttgcacc 73
                               ggt ggtgcaaaa  aattg ggtttttgccatt cttttaat gc    a aaa  g a  t cttt cacc
    humanchr1/239220001-239550000 302390 GGTCGGTGCAAAATCAATTGTGGTTTTTGCCATTGCTTTTAATTGCttttA-AAA--GTA-ATGCTTTTACACC 302459
                               899**********************************************955533.443..334.4689********* PP


                                  xxxxxxx RF
                  MADE1     74 aacctaa 80
                               aa ctaa
    humanchr1/239220001-239550000 302460 AATCTAA 302466
                               **99986 PP
```

Details of the alignment format are the same as for hmmsearch.

Finally, at the bottom of the file, you'll see some summary statistics. For example, at the bottom of the MADE1 search output, you'll find something like:

```
    Internal pipeline statistics summary:
    -------------------------------------
    Query model(s):                         1  (80 nodes)
    Target sequences:                       1  (660000 residues searched)
    Residues passing SSV filter:        61658  (0.0934); expected (0.02)
    Residues passing bias filter:       45802  (0.0694); expected (0.02)
    Residues passing Vit filter:         2443  (0.0037); expected (0.001)
    Residues passing Fwd filter:         2217  (0.00336); expected (1e-05)
    Total number of hits:                   5  (0.000403)
    # CPU time: 0.05u 0.00s 00:00:00.05 Elapsed: 00:00:00.03
    # Mc/sec: 1760.00
    //
```

This gives you some idea of what's going on in nhmmer's acceleration pipeline. You've got one query HMM, and 660,000 residues were searched (there are 330,000 bases in the single sequence found in the file; the search includes the reverse complement, doubling the search space). The sequences in the database go through a gauntlet of three scoring algorithms called SSV, Viterbi, and Forward, in order of increasing sensitivity and increasing computational requirement.

SSV (the "Single ungapped Segment Viterbi" algorithm) as used in nhmmer is closely related to the MSV algorithm used in hmmsearch, in that it depends on ungapped alignment segments. The difference lies in how those alignments are used. Using MSV, a sequence is either rejected or accepted in its entirety. In the scanning-SSV filter of nhmmer, each sequence in the database is scanned for high-scoring ungapped alignment segments, and a window around each such segment is extracted (merging overlapping windows),

and passed on to the next stage. By default, nhmmer is configured to allow sequence segments with a P-value of ≤ 0.02 through the SSV score filter (thus, if the database contained no homologs and P-values were accurately calculated, the highest scoring 2% of the sequence will pass the filter). Here, 61658 bases, or about 9% of the database, got through the SSV filter.

The quick "bias filter" is then applied, as in `hmmsearch`. Here, 45802 bases, roughly 7% of the database pass through the bias filter.

The Viterbi filter then calculates a gapped optimal alignment score for each window that survived the earlier stages. This score is a closer approximation than the SSV score of the final score that the window will achieve if it survives to final processing, but the Viterbi filter is about four-fold slower than SSV. By default, nhmmer lets windows with a P-value of ≤ 0.001 through this stage. Here, 2443 bases, about 0.4% of the database gets through.

Then the full Forward score is calculated, which sums over all possible alignments of the profile to the window. The default allows windows with a P-value of $\leq 10^{-5}$ through; 2217 bases passed.

All sequences that make it through these filters are then subjected to a full probabilistic analysis using the HMM Forward/Backward algorithms, to identify hit envelopes, then determine posterior probabilities for each aligned residue, followed by optimal accuracy alignment. The results of this step are what you finally see on the output. The final number of hits and fractional coverage of the database is shown next. This is typically smaller than the fraction of the database passing the Forward filter, as hit identification typically trims windows down to a smaller envelope.

Finally, nhmmer reports the speed of the search in units of Mc/sec (million dynamic programming cells per second), the CPU time, and the elapsed time. This search took about 0.03 seconds of elapsed (wall clock time).

There is not currently a DNA analog to `jackhmmer`.

Searching a profile HMM database with a query sequence

In some cases, rather than wishing to search a single model against a collection of sequences, you may wish to annotate all the instances of a collection of HMMs found in a single sequence.

In the case of proteins, `hmmscan` is for annotating all the different known/detectable domains in a given protein sequence. It takes a single query sequence and an HMM database as input. The HMM database might be Pfam, SMART, or TIGRFams, for example, or another collection of your choice.

In the case of DNA, the same purpose is met with `nhmmscan`. In this case, the HMM database might be Dfam (a database of HMMs for transposable element families), or a collection of conserved regulatory elements.

Here, we show an example of using `hmmscan`, which, you will see, produces output very much like that of `hmmsearch`. We omit details of running `nhmmscan` - it is run in the same way as `hmmscan`, and its output matches that of `nhmmer`.

Step 1: create an HMM database flatfile

An HMM "database" flatfile is simply a concatenation of individual HMM files. To create a database flatfile, you can either build individual HMM files and concatenate them, or you can concatenate Stockholm alignments and use `hmmbuild` to build an HMM database of all of them in one command.

Let's create a tiny database called `minifam` containing models of globin, fn3, and Pkinase (protein kinase) domains by concatenating model files:

> **hmmbuild globins4.hmm tutorial/globins4.sto**

> **hmmbuild fn3.hmm tutorial/fn3.sto**

> **hmmbuild Pkinase.hmm tutorial/Pkinase.sto**

> **cat globins4.hmm fn3.hmm Pkinase.hmm > minifam**

We'll use `minifam` for our examples in just a bit, but first a few words on other ways to build HMM databases, especially big ones. The file `tutorials/minifam` is the same thing, if you want to just use that.

Alternatively, you can concatenate Stockholm alignment files together (as Pfam does in its big `Pfam-A.seed` and `Pfam-A.full` flatfiles) and use `hmmbuild` to build HMMs for all the alignments at once. This won't work properly for our tutorial alignments, because the `globins4.sto` alignment doesn't have an `#=GF ID` annotation line giving a name to the globins4 alignment, so `hmmbuild` wouldn't know how to name it correctly. To build a multi-model database from a multi-MSA flatfile, the alignments have to be in Stockholm format (no other MSA format that I'm aware of supports having more than one alignment per file), and each alignment must have a name on a `#=GF ID` line.

But if you happen to have a Pfam seed alignment flatfile `Pfam-A.seed` around, an example command would be:

> **hmmbuild Pfam-A.hmm Pfam-A.seed**

This would take about two or three hours to build all 10,000 models or so in Pfam. To speed the database construction process up, `hmmbuild` supports MPI parallelization.

As far as HMMER's concerned, all you have to do is add `--mpi` to the command line for `hmmbuild`, assuming you've compiled support for MPI into it (see the installation instructions). You'll also need to know how to invoke an MPI job in your particular environment, with your job scheduler and MPI distribution. We can't really help you with this – different sites have different cluster environments.

With our scheduler (SGE, the Sun Grid Engine) and our MPI distro (Intel MPI), an example incantation for building `Pfam.hmm` from `Pfam-A.seed` is:

> **qsub -N hmmbuild -j y -o errors.out -b y -cwd -V -pe impi 128**
'mpirun -np 128 ./hmmbuild --mpi Pfam.hmm Pfam-A.seed > hmmbuild.out'

This reduces the time to build all of Pfam to about 40 seconds.

Step 2: compress and index the flatfile with hmmpress

The `hmmscan` program has to read a lot of profile HMMs in a hurry, and HMMER's ASCII flatfiles are bulky. To accelerate this, `hmmscan` uses binary compression and indexing of the flatfiles. To use `hmmscan`, you must first compress and index your HMM database with the `hmmpress` program:

> **hmmpress minifam**

This will quickly produce:

```
Working...    done.
Pressed and indexed 3 HMMs (3 names and 2 accessions).
Models pressed into binary file:   minifam.h3m
SSI index for binary model file:   minifam.h3i
Profiles (MSV part) pressed into:  minifam.h3f
Profiles (remainder) pressed into: minifam.h3p
```

and you'll see these four new binary files in the directory.

The `tutorial/minifam` example has already been pressed, so there are example binary files `tutorial/minifam.h3{m,i,f,p}` included in the tutorial.

Their format is "proprietary", which is an open source term of art that means both "I haven't found time to document them yet" and "I still might decide to change them arbitrarily without telling you".

Step 3: search the HMM database with hmmscan

Now we can analyze sequences using our HMM database and `hmmscan`.

For example, the receptor tyrosine kinase 7LESS_DROME not only has all those fibronectin type III domains on its extracellular face, it's got a protein kinase domain on its intracellular face. Our `minifam` database has models of both `fn3` and `Pkinase`, as well as the unrelated `globins4` model. So what happens when we scan the 7LESS_DROME sequence:

> **hmmscan minifam tutorial/7LESS_DROME**

The header and the first section of the output will look like:

```
# hmmscan :: search sequence(s) against a profile database
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# query sequence file:             7LESS_DROME
# target HMM database:             minifam
# per-seq hits tabular output:     7LESS.tbl
# per-dom hits tabular output:     7LESS.domtbl
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query:       7LESS_DROME  [L=2554]
Accession:   P13368
Description: RecName: Full=Protein sevenless;        EC=2.7.10.1;
Scores for complete sequence (score includes all domains):
   --- full sequence ---   --- best 1 domain ---    -#dom-
    E-value  score  bias    E-value  score  bias    exp  N  Model      Description
    ------- ------ -----    ------- ------ -----    ---- --  --------   -----------
    5.6e-57  178.0   0.4    3.5e-16   47.2   0.9     9.4  9  fn3        Fibronectin type III domain
    1.1e-43  137.2   0.0    1.7e-43  136.5   0.0     1.3  1  Pkinase    Protein kinase domain
```

The output fields are in the same order and have the same meaning as in hmmsearch's output.

The size of the search space for hmmscan is the number of models in the HMM database (here, 3; for a Pfam search, on the order of 10000). In hmmsearch, the size of the search space is the number of sequences in the sequence database. This means that E-values may differ even for the same individual profile vs. sequence comparison, depending on how you do the search.

For domain, there then follows a domain table and alignment output, just as in hmmsearch. The fn3 annotation, for example, looks like:

```
Domain annotation for each model (and alignments):
>> fn3  Fibronectin type III domain
   #     score  bias  c-Evalue  i-Evalue hmmfrom  hmm to    alifrom  ali to    envfrom  env to     acc
 ---   ------ ----- --------- --------- ------- -------    ------- -------    ------- -------    ----
   1 ?   -1.3   0.0      0.33       0.5      61      74 ..     396     409 ..     395     411 .. 0.85
   2 !   40.7   0.0   2.6e-14   3.8e-14       2      84 ..     439     520 ..     437     521 .. 0.95
   3 !   14.4   0.0   4.1e-06   6.1e-06      13      85 ..     836     913 ..     826     914 .. 0.73
   4 !    5.1   0.0    0.0032    0.0048      10      36 ..    1209    1235 ..    1203    1259 .. 0.82
   5 !   24.3   0.0   3.4e-09     5e-09      14      80 ..    1313    1380 ..    1304    1386 .. 0.82
   6 ?    0.0   0.0      0.13      0.19      58      72 ..    1754    1768 ..    1739    1769 .. 0.89
   7 !   47.2   0.9   2.3e-16   3.5e-16       1      85 [.    1799    1890 ..    1799    1891 .. 0.91
   8 !   17.8   0.0   3.7e-07   5.5e-07       6      74 ..    1904    1966 ..    1901    1976 .. 0.90
   9 !   12.8   0.0   1.3e-05     2e-05       1      86 []    1993    2107 ..    1993    2107 .. 0.89
```

and an example alignment (of that second domain again):

```
== domain 2  score: 40.7 bits;  conditional E-value: 2.6e-14
               ---CEEEEEEECTTEEEEEEE--S--SS--SEEEEEEEETTTCCGCEEEEEETTTSEEEEES--TT-EEEEEEEEEETTEE-E CS
        fn3   2 saPenlsvsevtstsltlsWsppkdgggpitgYeveyqekgegeewqevtvprtttsvtltgLepgteYefrVqavngagegp 84
                saP ++ +  ++ l ++W p +  +gpi+gY++++++++++  + e+ vp+   s+ +++L++gt+Y++ +  +n++gegp
7LESS_DROME 439 SAPVIEHLMGLDDSHLAVHWHPGRFTNGPIEGYRLRLSSSEGNA-TSEQLVPAGRGSYIFSQLQAGTNYTLALSMINKQGEGP 520
                78999999999***************************9998.************************************9997 PP
```

You'd think that except for the E-values (which depend on database search space sizes), you should get exactly the same scores, domain number, domain coordinates, and alignment every time you do a search of the same HMM against the same sequence. And this is actually the case – but in fact, it's actually not so obvious this should be so, and HMMER is going out of its way to make it so. HMMER uses stochastic sampling algorithms to infer some parameters, and also to infer the exact domain number and domain boundaries in certain difficult cases. If HMMER ran its stochastic samples "properly", it would see different samples every time you ran a program, and all of you would complain to me that HMMER was weird and buggy because it gave different answers on the same problem. To suppress run-to-run variation, HMMER seeds its random number generator(s) identically every time you do a sequence comparison. If you're an expert, and you really want to see the proper stochastic variation that results from any sampling algorithms that got run, you can pass a command-line argument of --seed 0 to programs that have this property (hmmbuild and the four search programs).

35

Creating multiple alignments with hmmalign

The file `tutorial/globins45.fa` is a FASTA file containing 45 unaligned globin sequences. To align all of these to the `globins4` model and make a multiple sequence alignment:

> **`> hmmalign globins4.hmm tutorial/globins45.fa`**

The output of this is a Stockholm format multiple alignment file. The first few lines of it look like:

```
# STOCKHOLM 1.0

MYG_ESCGI            .-VLSDAEWQLVLNIWAKVEADVAGHGQDILIRLFKGHPETLEKFDKFKH
#=GR MYG_ESCGI  PP  ..69**************************************************
MYG_HORSE            g--LSDGEWQQVLNVWGKVEADIAGHGQEVLIRLFTGHPETLEKFDKFKH
#=GR MYG_HORSE  PP  8..89**************************************************
MYG_PROGU            g--LSDGEWQLVLNVWGKVEGDLSGHGQEVLIRLFKGHPETLEKFDKFKH
#=GR MYG_PROGU  PP  8..89**************************************************
MYG_SAISC           g--LSDGEWQLVLNIWGKVEADIPSHGQEVLISLFKGHPETLEKFDKFKH
#=GR MYG_SAISC  PP  8..89**************************************************
MYG_LYCPI           g--LSDGEWQIVLNIWGKVETDLAGHGQEVLIRLFKNHPETLDKFDKFKH
#=GR MYG_LYCPI  PP  8..89**************************************************
MYG_MOUSE           g--LSDGEWQLVLNVWGKVEADLAGHGQEVLIGLFKTHPETLDKFDKFKN
#=GR MYG_MOUSE  PP  8..89**************************************************
MYG_MUSAN           v------DWEKVNSVWSAVESDLTAIGQNILLRLFEQYPESQNHFPKFKN
...
```

and so on.

First thing to notice here is that `hmmalign` uses both lower case and upper case residues, and it uses two different characters for gaps. This is because there are two different kinds of columns: "match" columns in which residues are assigned to match states and gaps are treated as deletions relative to consensus, and "insert" columns where residues are assigned to insert states and gaps in other sequences are just padding for the alignment to accomodate those insertions. In a match column, residues are upper case, and a '-' character means a deletion relative to the consensus. In an insert column, residues are lower case, and a '.' is padding. A '-' deletion has a cost: transition probabilities were assessed, penalizing the transition into and out of a deletion. A '.' pad has no cost per se; instead, the sequence(s) with insertions are paying transition probabilities into and out of their inserted residue.

This notation is only for your convenience in output files: you can see the structure of the profile HMM reflected in the pattern of residues and gap characters [7]. In input files, in most alignment formats[8] HMMER is case-insensitive, and it does not distinguish between different gap characters: '-' (dash), '.' (period), or even '_' (underscore) are accepted as gap characters.

Important: insertions in a profile HMM are *unaligned*. Suppose one sequence has an insertion of length 10 and another has an insertion of length 2 in the same place in the profile. The alignment will show ten insert columns, to accomodate the longest insertion. The residues of the shorter insertion are thrown down in an arbitrary order. (If you must know: by arbitrary HMMER convention, the insertion is divided in half; half is left-justified, and the other half is right-justified, leaving '.' characters in the middle.) Notice that in the previous paragraph we oh-so-carefully said residues are "assigned" to a state, not "aligned". For match states, assigned and aligned are the same thing: a one-to-one correspondence between a residue and a consensus match state in the model. But there may be one *or more* residues assigned to the same insert state.

Don't be confused by the unaligned nature of profile HMM insertions. You're sure to see cases where lower-case inserted residues are "obviously misaligned". This is just because HMMER isn't trying to "align" them in the first place: it is assigning them to unaligned insertions.

Enough about the sequences in the alignment. Now notice all those `PP` annotation lines. That's posterior probability annotation, as in the single sequence alignments that `hmmscan` and `hmmsearch` showed. This essentially represents the confidence that each residue is assigned where it should be.

---

[7]By default, `hmmalign` removes any columns that are all deletion characters, so the number of apparent match columns in a displayed alignment is $\leq$ the actual number of match states in the profile. To prevent this trimming and see columns for all match states, use the `--allcol` option. This can be helpful if you're writing some postprocessor that's trying to keep track of what columns are assigned to what match states in the profile.

[8]A2M format is the exception.

Again, that's "assigned", not "aligned". The posterior probability assigned to an inserted residue is the probability that it is assigned to the insert state that corresponds to that column. Because the same insert state might correspond to more than one column, the probability on an insert residue is *not* the probability that it belongs in that particular column; again, where there's a choice of column for inserted residues, that choice is arbitrary.

The program hmmalign currently has a "feature" that we're aware of. Recall that HMMER only does local alignments. Here, we know that we've provided full length globin sequences, and globins4 is a full length globin model. We'd probably like hmmalign to produce a global alignment. It can't currently do that. If it doesn't quite manage to extend its local alignment to the full length of a target globin sequence, you'll get a weird-looking effect, as the nonmatching termini are pulled out to the left or right. For example, look at the N-terminal g in MYG_HORSE above. HMMER is about 80% confident that this residue is nonhomologous, though any sensible person would align it into the first globin consensus column.

Look at the end of that first block of Stockholm alignment, where you'll see:

```
...
HBBL_RANCA           v-HWTAEEKAVINSVWQKV--DVEQDGHEALTRLFIVYPWTQRYFSTFGD
#=GR HBBL_RANCA PP   6.6799*************..****************************
HBB2_TRICR           .VHLTAEDRKEIAAILGKV--NVDSLGGQCLARLIVVNPWSRRYFHDFGD
#=GR HBB2_TRICR PP   .69**************..****************************
#=GC PP_cons         .679*********************************************
#=GC RF              .xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

The #=GC PP_cons line is Stockholm-format *consensus posterior probability* annotation for the entire column. It's calculated simply as the arithmetic mean of the per-residue posterior probabilities in that column. This should prove useful in phylogenetic inference applications, for example, where it's common to mask away nonconfidently aligned columns of a multiple alignment. The PP_cons line provides an objective measure of the confidence assigned to each column.

The #=GC RF line is Stockholm-format *reference coordinate annotation*, with an x marking each column that the profile considered to be consensus.

4   The HMMER profile/sequence comparison pipeline

In this section, we briefly outline the processing pipeline for a single profile/sequence comparison.[1] This should help give you a sense of what HMMER is doing under the hood, what sort of mistakes it may make (rarely, of course!), and what the various results in the output actually mean. We'll first describe the pipeline in the context of protein search (`phmmer`, `hmmsearch`, `hmmscan`, `jackhmmer`), then wrap back around to discuss the modified pipeline used in `nhmmer` and `nhmmscan`.

In briefest outline, the comparison pipeline takes the following steps:

Null model. Calculate a score term for the "null hypothesis" (a probability model of *non*-homology). This score correction is used to turn all subsequent profile/sequence bit scores into a final log-odds bit score.

MSV filter. The main acceleration heuristic. The MSV ("Multiple Segment Viterbi") algorithm looks for one or more high-scoring *ungapped* alignments. If the MSV score passes a set threshold, the entire sequence passes on to the next pipeline step; else it is rejected.

Bias filter. A hack that reduces false positive MSV hits due to biased composition sequences. A two-state HMM is constructed from the mean residue composition of the profile and the standard residue composition of the null model, and used to score the sequence. The MSV bit score is corrected using this as a second null hypothesis. If the MSV score still passes the MSV threshold, the sequence passes on to the next step; else it is rejected. The bias filter score correction will also be applied to the Viterbi filter and Forward filter scores that follow.

Viterbi filter. A more stringent accelerated filter. An optimal (maximum likelihood) gapped alignment score is calculated. If this score passes a set threshold, the sequence passes to the next step; else it is rejected.

Forward filter/parser. The full likelihood of the profile/sequence comparison is evaluated, summed over the entire alignment ensemble, using the HMM Forward algorithm. This score is corrected to a bit score using the null model and bias filter scores. If the bit score passes a set threshold, the sequence passes on to the next step; else it is rejected.

Domain identification. Using the Forward parser results, now combined with a Backward parser, posterior probabilities of domain locations are calculated. A discrete set of putative domains (alignments) is identified by applying heuristics to posterior probabilities. This procedure identifies *envelopes*: subsequences on the target sequence which contain a lot of probability mass for a match to the profile.

Alignment. For each identified domain, a full Forward/Backward algorithm is performed. An *ad hoc* "null2" hypothesis is constructed for each domain's composition and used to calculate a biased composition score correction. A maximum expected accuracy (MEA) alignment is calculated. This identifies one MEA alignment within each envelope.

Storage. Now we have a *sequence score* (and P-value); the sequence contains one or more domains, each of which has a *domain score* (and P-value), and each domain has an MEA alignment annotated with per-residue posterior probabilities.

In more detail, each step is described in subsections that follow. Even more detail may be found in (Eddy, 2011).

---

[1]Code gurus and masochists: you can follow along in `src/p7_pipeline.c`.

Null model.

The "null model" calculates the probability that the target sequence is *not* homologous to the query profile. A HMMER bit score is the log of the ratio of the sequence's probability according to the profile (the homology hypothesis) over the null model probability (the non-homology hypothesis).

The null model is a one-state HMM configured to generate "random" sequences of the same mean length L as the target sequence, with each residue drawn from a background frequency distribution (a standard i.i.d. model: residues are treated as independent and identically distributed). Currently, this background frequency distribution is hardcoded as the mean residue frequencies in Swiss-Prot 50.8 (October 2006).

For technical reasons, HMMER incorporates the *residue emission* probabilities of the null model directly into the profile, by turning each emission probability in the profile into an odds ratio. The null model score calculation therefore is only concerned with accounting for the remaining *transition* probabilities of the null model and toting them up into a bit score correction. The null model calculation is fast, because it only depends on the length of the target sequence, not its sequence.

MSV filter.

The sequence is aligned to the profile using a specialized model that allows multiple high-scoring local ungapped segments to match. The optimal alignment score (Viterbi score) is calculated under this multi-segment model, hence the term MSV, for "multi-segment Viterbi". This is HMMER's main speed heuristic.

The MSV score is comparable to BLAST's sum score (optimal sum of ungapped alignment segments). Roughly speaking, MSV is comparable to skipping the heuristic word hit and hit extension steps of the BLAST acceleration algorithm.

The MSV filter is very, very fast. In addition to avoiding indel calculations in the dynamic programming table, it uses reduced precision scores scaled to 8-bit integers, enabling acceleration via 16-way parallel SIMD vector instructions.

The MSV score is a true log-odds likelihood ratio, so it obeys conjectures about the expected score distribution (Eddy, 2008) that allow immediate and accurate calculation of the statistical significance (P-value) of the MSV bit score.

By default, comparisons with a P-value of ≤ 0.02 pass this filter, meaning that about 2% of nonhomologous sequences are expected to pass. You can use the `--F1 <x>` option to change this threshold. For example, `--F1 <0.05>` would pass 5% of the comparisons, making a search more sensitive but slower. Setting the threshold to ≥ 1.0 (`--F1 99` for example) assures that all comparisons will pass. Shutting off the MSV filter may be worthwhile if you want to make sure you don't miss comparisons that have a lot of scattered insertions and deletions. Alternatively, the `--max` option causes the MSV filter step (and all other filter steps) to be bypassed.

The MSV bit score is calculated as a log-odds score using the null model for comparison. No correction for a biased composition or repetitive sequence is done at this stage. For comparisons involving biased sequences and/or profiles, more than 2% of comparisons will pass the MSV filter. At the end of search output, there is a line like:

```
Passed MSV filter:                    107917  (0.020272); expected 106468.8 (0.02)
```

which tells you how many and what fraction of comparisons passed the MSV filter, versus how many (and what fraction) were expected.

Biased composition filter.

It's possible for profiles and/or sequences to have biased residue compositions that result in "significant" log-odds bit scores not because the profile matches the sequence particularly well, but because the *null model* matches the sequence particularly badly.

HMMER uses fairly good methods to compensate its scores for biased composition, but these methods are computationally expensive and applied late in the pipeline (described below).

In a few cases, profiles and/or target sequences are sufficiently biased that too many comparisons pass the MSV filter, causing HMMER speed performance to be severely degraded. Although the final scores and E-values at the end of the pipeline will be calculated taking into account a "null2" model of biased composition and simple repetition, the null2 model is dependent on a full alignment ensemble calculation via the Forward/Backward algorithm, making it computationally complex, so it won't get calculated until the very end. The treatment of biased composition comparisons is probably the most serious problem remaining in HMMER. Solving it well will require more research. As a stopgap solution to rescuing most of the speed degradation while not sacrificing too much sensitivity, an *ad hoc* biased composition filtering step is applied to remove highly biased comparisons.

On the fly, a two-state HMM is constructed. One state emits residues from the background frequency distribution (same as the null1 model), and the other state emits residues from the mean residue composition of the profile (i.e. the expected composition of sequences generated by the core model, including match and insert states [`p7_hmm.c:p7_hmm_SetComposition()`]). Thus if the profile is highly biased (cysteine-rich, for example; or highly hydrophobic with many transmembrane segments), this composition bias will be captured by this second state. This model's transitions are arbitrarily set such that state 1 emits an expected length of 400 at a time, and state 2 emits an expected length of M/8 at a time (for a profile of length M). An overall target sequence length distribution is set to a mean of L, identical to the null1 model.

The sequence is then rescored using this "bias filter model" in place of the null1 model, using the HMM Forward algorithm. (This replaces the null1 model score at all subsequent filter steps in the pipeline, until a final Forward score is calculated.) A new MSV bit score is obtained.

If the P-value of this still satisfies the MSV thresholds, the sequence passes the biased composition filter.

The `--F1 <x>` option controls the P-value threshold for passing the MSV filter score, both before (with the simple null1 model) and after the bias composition filter is applied.

The `--max` option bypasses all filters in the pipeline, including the bias filter.

The `--nobias` option turns off (bypasses) the biased composition filter. The simple null model is used as a null hypothesis for MSV and in subsequent filter steps. The biased composition filter step compromises a small amount of sensitivity. Though it is good to have it on by default, you may want to shut it off if you know you will have no problem with biased composition hits.

At the end of a search output, you will see a line like:

```
Passed bias filter:                    105665  (0.019849); expected 106468.8 (0.02)
```

which tells you how many and what fraction of comparisons passed the biased composition filter, versus how many were expected. (If the filter was turned off, all comparisons pass.)

Viterbi filter.

The sequence is now aligned to the profile using a fast Viterbi algorithm for optimal gapped alignment.

This Viterbi implementation is specialized for speed. It is implemented in 8-way parallel SIMD vector instructions, using reduced precision scores that have been scaled to 16-bit integers. Only one row of the dynamic programming matrix is stored, so the routine only recovers the score, not the optimal alignment itself. The reduced representation has limited range; local alignment scores will not underflow, but high scoring comparisons can overflow and return infinity, in which case they automatically pass the filter.

The final Viterbi filter bit score is then computed using the appropriate null model log likelihood (by default the biased composition filter model score, or if the biased filter is off, just the null model score). If the P-value of this score passes the Viterbi filter threshold, the sequence passes on to the next step of the pipeline.

The `--F2 <x>` option controls the P-value threshold for passing the Viterbi filter score. The default is 0.001. The `--max` option bypasses all filters in the pipeline.

At the end of a search output, you will see a line like:

```
      Passed Vit filter:                    2207   (0.00443803); expected 497.3 (0.001)
```

which tells you how many and what fraction of comparisons passed the Viterbi filter, versus how many were expected.

Forward filter/parser.

The sequence is now aligned to the profile using the full Forward algorithm, which calculates the likelihood of the target sequence given the profile, summed over the ensemble of all possible alignments.

This is a specialized time- and memory-efficient Forward implementation called the "Forward parser". It is implemented in 4-way parallel SIMD vector instructions, in full precision (32-bit floating point). It stores just enough information that, in combination with the results of the Backward parser (below), posterior probabilities of start and stop points of alignments (domains) can be calculated in the domain definition step (below), although the detailed alignments themselves cannot be.

The Forward filter bit score is calculated by correcting this score using the appropriate null model log likelihood (by default the biased composition filter model score, or if the biased filter is off, just the null model score). If the P-value of this bit score passes the Forward filter threshold, the sequence passes on to the next step of the pipeline.

The bias filter score has no further effect in the pipeline. It is only used in filter stages. It has *no* effect on final reported bit scores or P-values. Biased composition compensation for final bit scores is done by a more complex domain-specific algorithm, described below.

The `--F3 <x>` option controls the P-value threshold for passing the Forward filter score. The default is 1e-5. The `--max` option bypasses all filters in the pipeline.

At the end of a search output, you will see a line like:

```
      Passed Fwd filter:                    1076   (0.00216371); expected 5.0 (1e-05)
```

which tells you how many and what fraction of comparisons passed the Forward filter, versus how many were expected.

Domain definition.

A target sequence that reaches this point is very likely to contain one or more significant matches to the profile. These matches are referred to as "domains", since the main use of HMMER has historically been to match profile HMMs from protein domain databases like Pfam, and one of HMMER's strengths is to be able to cleanly parse a multidomain target sequence into its multiple nonoverlapping hits to the same domain model.

The domain definition step is essentially its own pipeline, with steps as follows:[2]

Backward parser.   The counterpart of the Forward parser algorithm is calculated in an analogous time- and memory-efficient implementation. The Forward algorithm gives the likelihood of all *prefixes* of the target sequence, summed over their alignment ensemble, and the Backward algorithm gives the likelihood of all *suffixes*. For any given point of a possible model state/residue alignment, the product of the Forward and Backward likelihoods gives the likelihood of the entire alignment ensemble conditional on using that particular alignment point. Thus, we can calculate things like the posterior probability that an alignment starts or ends at a given position in the target sequence.

---

[2]Code gurus and masochists can follow along in `src/p7_domaindef.c`.

Domain decoding.   The posterior decoding algorithm is applied, to calculate the posterior probability of alignment starts and ends (profile B and E state alignments) with respect to target sequence position.

The sum of the posterior probabilities of alignment starts (B states) over the entire target sequence is the *expected number of domains* in the sequence.

In a tabular output (`--tblout`) file, this number is in the column labeled `exp`.

Region identification.   A heuristic is now applied to identify a *non-overlapping* set of "regions" that contain significant probability mass suggesting the presence of a match (alignment) to the profile.

For each region, the expected number of domains is calculated (again by posterior decoding on the Forward/Backward parser results). This number should be about 1: we expect each region to contain one local alignment to the profile.

In a tabular output (`--tblout`) file, the number of discrete regions identified by this posterior decoding step is in the column labeled `reg`. It ought to be almost the same as the expectation `exp`. If it is not, there may be something funny going on, like a tandem repetitive element in the target sequence (which can produce so many overlapping weak hits that the sequence appears to be a significant hit with lots of domains expected *somewhere*, but the probability is fuzzed out over the repetitive region and few or no good discrete alignment regions can be identified).

Envelope identification.   Now, within each region, we will attempt to identify *envelopes*. An *envelope* is a subsequence of the target sequence that appears to contain alignment probability mass for a likely domain (one local alignment to the profile).

When the region contains $\simeq 1$ expected domain, envelope identification is already done: the region's start and end points are converted directly to the envelope coordinates of a putative domain.

There are a few cases where the region appears to contain more than one expected domain – where more than one domain is closely spaced on the target sequence and/or the domain scores are weak and the probability masses are ill-resolved from each other. These "multidomain regions", when they occur, are passed off to an even more *ad hoc* resolution algorithm called *stochastic traceback clustering*. In stochastic traceback clustering, we sample many alignments from the posterior alignment ensemble, cluster those alignments according to their overlap in start/end coordinates, and pick clusters that sum up to sufficiently high probability. Consensus start and end points are chosen for each cluster of sampled alignments. These start/end points define envelopes.

These envelopes identified by stochastic traceback clustering are *not* guaranteed to be nonoverlapping. It's possible that there are alternative "solutions" for parsing the sequence into domains, when the correct parsing is ambiguous. HMMER will report all high-likelihood solutions, not just a single nonoverlapping parse.

It's also possible (though rare) for stochastic clustering to identify *no* envelopes in the region.

In a tabular output (`--tblout`) file, the number of regions that had to be subjected to stochastic traceback clustering is given in the column labeled `clu`. This ought to be a small number (often it's zero). The number of envelopes identified by stochastic traceback clustering that overlap with other envelopes is in the column labeled `ov`. If this number is non-zero, you need to be careful when you interpret the details of alignments in the output, because HMMER is going to be showing overlapping alternative solutions. The total number of domain envelopes identified (either by the simple method or by stochastic traceback clustering) is in the column labeled `env`. It ought to be almost the same as the expectation and the number of regions.

Maximum expected accuracy alignment.   Each envelope is now aligned to the profile using the full Forward/Backward algorithm. The profile is configured to "unihit" mode, so that the profile expects only one local alignment (domain) in the envelope (as opposed to multiple domains). Posterior decoding is used to calculate the posterior probability of every detailed alignment of profile state to sequence residue. The posterior decodings are used to extract a "maximum expected accuracy" alignment. Each aligned residue is annotated with its posterior probability in the Forward/Backward alignment ensemble.

Currently, the Forward, Backward, and posterior decoding calculations at this step are *not* memory efficient. They calculate matrices requiring roughly $36ML$ bytes, where M is the profile length and L is the length of the envelope subsequence. Usually in `hmmsearch` and `hmmscan`, profiles and envelopes are small enough that this is not a problem. For example, a typical Pfam domain model is about 200 residues long, matching to individual target envelopes of about 200 residues each; this requires about 1.4 MB of memory in MEA alignment. However, in `phmmer` and `jackhmmer` programs, it's often going to be the case that you're aligning an entire query sequence to an entire target sequence in a single unresolved "domain" alignment. If this is titin (about 40,000 residues), it would require 57.6 GB of RAM. For this reason, currently, `phmmer` and `jackhmmer` can only handle query sequences of up to a few thousand residues. If you see a "fatal exception" error complaining about failure of a large memory allocation, you're almost certainly seeing a prohibitive memory requirement at this stage.[3]

In a tabular output (`--tblout`) file, the number of domains in envelopes (before any significance thresholding) is in the column labeled `dom`. This will generally be the same as the number of envelopes.

Biased composition score correction ("null2")   An *ad hoc* biased composition score correction is calculated for each envelope, using the posterior decoding. A corrected bit score and P-value for each envelope is calculated. These null2-corrected scores are subjected to the reporting and inclusion thresholds, at both the full sequence level and per-domain.

Modifications to the pipeline as used for DNA search.

SSV, not MSV.

In the MSV filter, one or more high-scoring ungapped segments contribute to a score that, if sufficiently high, casues the entire sequence to be passed on to the next stage (the bias filter). This strategy won't work for long DNA sequences; it doesn't filter the human genome much to say "there's a hit on chromosome 1, now process the whole thing". In the scanning-SSV ("Single ungapped Segment Viterbi") algorithm used in `nhmmer` and `nhmmscan`, each comparison between a query and target is scanned for high-scoring ungapped alignment segments, and a window around each such segment is extracted, merging overlapping windows. Each window is then passed on to the remaining filter cascade, where it is (for the most part) treated as described above. As with the MSV filter, the default P-value threshold is 0.02, and can be controlled with the `--F1` flag.

The `--max` flag also controls the amount of the sequence database that passes the SSV filter, but instead of the threshold being set to 1.0, as described for the protein pipeline, it is set to 0.4.

There are no domains, but there are envelopes.

In HMMER's protein-search programs, multiple matches of the model to a target sequence are treated as domains contained within a single hit for that sequence. In the DNA-search programs, each match of the model to a subsequence is treated as an independent hit - there's no notion of a domain. This is largely a difference in reporting; both pipelines rely on essentially the same envelope detection code; envelopes lead to domains in protein search, and hits in DNA search.

Biased composition.

DNA sequence is littered with regions containing tandem simple repeats or other low complexity sequence. Without accounting for such composition bias, we see many cases in which one part of a hit is obviously legitimate, and serves as the anchor for a neighboring alignment segment that is clearly low-complexity garbage, one form of a problem known as homologous overextension (Gonzalez and Pearson, 2010). The

---

[3]We know how to fix this, with memory-efficient algorithms, and are working on it.

null2 method used in protein search delays score modification until after the alignment is complete, but we know that this kind of overextension can be (mostly) avoided if the model's log odds scores account for the composition bias of the target region while constructing the alignment. The DNA search pipeline therefore does just this: it modifies the scoring scheme for each target envelope as a function of that envelope's sequence composition, then builds the alignment according to that scheme.

5 Tabular output formats

The target hits table

The `--tblout` output option produces the *target hits table*. The target hits table consists of one line for each different query/target comparison that met the reporting thresholds, ranked by decreasing statistical significance (increasing E-value).

tblout fields for protein search programs In the protein search programs, each line consists of **18 space-delimited fields** followed by a free text target sequence description, as follows:[1]

(1) `target name`: The name of the target sequence or profile.

(2) `accession`: The accession of the target sequence or profile, or '-' if none.

(3) `query name`: The name of the query sequence or profile.

(4) `accession`: The accession of the query sequence or profile, or '-' if none.

(5) `E-value (full sequence)`: The expectation value (statistical significance) of the target. This is a *per query* E-value; i.e. calculated as the expected number of false positives achieving this comparison's score for a *single* query against the Z sequences in the target dataset. If you search with multiple queries and if you want to control the *overall* false positive rate of that search rather than the false positive rate per query, you will want to multiply this per-query E-value by how many queries you're doing.

(6) `score (full sequence)`: The score (in bits) for this target/query comparison. It includes the biased-composition correction (the "null2" model).

(7) `Bias (full sequence)`: The biased-composition correction: the bit score difference contributed by the null2 model. High bias scores may be a red flag for a false positive, especially when the bias score is as large or larger than the overall bit score. It is difficult to correct for all possible ways in which a nonrandom but nonhomologous biological sequences can appear to be similar, such as short-period tandem repeats, so there are cases where the bias correction is not strong enough (creating false positives).

(8) `E-value (best 1 domain)`: The E-value if only the single best-scoring domain envelope were found in the sequence, and none of the others. If this E-value isn't good, but the full sequence E-value is good, this is a potential red flag. Weak hits, none of which are good enough on their own, are summing up to lift the sequence up to a high score. Whether this is Good or Bad is not clear; the sequence may contain several weak homologous domains, or it might contain a repetitive sequence that is hitting by chance (i.e. once one repeat hits, all the repeats hit).

(9) `score (best 1 domain)`: The bit score if only the single best-scoring domain envelope were found in the sequence, and none of the others. (Inclusive of the null2 bias correction.]

(10) `bias (best 1 domain)`: The null2 bias correction that was applied to the bit score of the single best-scoring domain.

(11) `exp`: Expected number of domains, as calculated by posterior decoding on the mean number of begin states used in the alignment ensemble.

---

[1]The `tblout` format is deliberately space-delimited (rather than tab-delimited) and justified into aligned columns, so these files are suitable both for automated parsing and for human examination. Tab-delimited data files are difficult for humans to examine and spot check. For this reason, we think tab-delimited files are a minor evil in the world. Although we occasionally receive shrieks of outrage about this, we stubbornly feel that space-delimited files are just as trivial to parse as tab-delimited files.

(12) `reg`: Number of discrete regions defined, as calculated by heuristics applied to posterior decoding of begin/end state positions in the alignment ensemble. The number of regions will generally be close to the expected number of domains. The more different the two numbers are, the less discrete the regions appear to be, in terms of probability mass. This usually means one of two things. On the one hand, weak homologous domains may be difficult for the heuristics to identify clearly. On the other hand, repetitive sequence may appear to have a high expected domain number (from lots of crappy possible alignments in the ensemble, no one of which is very convincing on its own, so no one region is discretely well-defined).

(13) `clu`: Number of regions that appeared to be multidomain, and therefore were passed to stochastic traceback clustering for further resolution down to one or more envelopes. This number is often zero.

(14) `ov`: For envelopes that were defined by stochastic traceback clustering, how many of them overlap other envelopes.

(15) `env`: The total number of envelopes defined, both by single envelope regions and by stochastic traceback clustering into one or more envelopes per region.

(16) `dom`: Number of domains defined. In general, this is the same as the number of envelopes: for each envelope, we find an MEA (maximum expected accuracy) alignment, which defines the endpoints of the alignable domain.

(17) `rep`: Number of domains satisfying reporting thresholds. If you've also saved a `--domtblout` file, there will be one line in it for each reported domain.

(18) `inc`: Number of domains satisfying inclusion thresholds.

(19) `description of target`: The remainder of the line is the target's description line, as free text.

tblout fields for DNA search programs In the DNA search programs, there is less concentration on domains, and more focus on presenting the hit ranges. Each line consists of **15 space-delimited fields** followed by a free text target sequence description, as follows:

(1) `target name`: The name of the target sequence or profile.

(2) `accession`: The accession of the target sequence or profile, or '-' if none.

(3) `query name`: The name of the query sequence or profile.

(4) `accession`: The accession of the query sequence or profile, or '-' if none.

(5) `hmmfrom`: The position in the hmm at which the hit starts.

(6) `hmm to`: The position in the hmm at which the hit ends.

(7) `alifrom`: The position in the target sequence at which the hit starts.

(8) `ali to`: The position in the target sequence at which the hit ends.

(9) `envfrom`: The position in the target sequence at which the surrounding envelope starts.

(10) `env to`: The position in the target sequence at which the surrounding envelope ends.

(11) `sq len`: The length of the target sequence..

(12) `strand`: The strand on which the hit was found ("-" when alifrom¿ali to).

(13) `E-value`: The expectation value (statistical significance) of the target, as above.

(14) `score (full sequence):` The score (in bits) for this hit. It includes the biased-composition correction.

(15) `Bias (full sequence):` The biased-composition correction, as above

(16) `description of target:` The remainder of the line is the target's description line, as free text.

These tables are columnated neatly for human readability, but do not write parsers that rely on this columnation; rely on space-delimited fields. The pretty columnation assumes fixed maximum widths for each field. If a field exceeds its allotted width, it will still be fully represented and space-delimited, but the columnation will be disrupted on the rest of the row.

Note the use of target and query columns. A program like `hmmsearch` searches a query profile against a target sequence database. In an `hmmsearch` tblout file, the sequence (target) name is first, and the profile (query) name is second. A program like `hmmscan`, on the other hand, searches a query sequence against a target profile database. In a `hmmscan` tblout file, the profile name is first, and the sequence name is second. You might say, hey, wouldn't it be more consistent to put the profile name first and the sequence name second (or vice versa), so `hmmsearch` and `hmmscan` tblout files were identical? Well, first of all, they still wouldn't be identical, because the target database size used for E-value calculations is different (number of target sequences for `hmmsearch`, number of target profiles for `hmmscan`, and it's good not to forget this. Second, what about programs like `phmmer` where the query is a sequence and the targets are also sequences?

If the "domain number estimation" section of the protein table (exp, reg, clu, ov, env, dom, rep, inc) makes no sense to you, it may help to read the previous section of the manual, which describes the HMMER processing pipeline, including the steps that probabilistically define domain locations in a sequence.

The domain hits table (protein search only)

In protein search programs, the `--domtblout` option produces the *domain hits table*. There is one line for each domain. There may be more than one domain per sequence. The domain table has **22 whitespace-delimited fields** followed by a free text target sequence description, as follows:

(1) `target name:` The name of the target sequence or profile.

(2) `target accession:` Accession of the target sequence or profile, or '-' if none is available.

(3) `tlen:` Length of the target sequence or profile, in residues. This (together with the query length) is useful for interpreting where the domain coordinates (in subsequent columns) lie in the sequence.

(4) `query name:` Name of the query sequence or profile.

(5) `accession:` Accession of the target sequence or profile, or '-' if none is available.

(6) `qlen:` Length of the query sequence or profile, in residues.

(7) `E-value:` E-value of the overall sequence/profile comparison (including all domains).

(8) `score:` Bit score of the overall sequence/profile comparison (including all domains), inclusive of a null2 bias composition correction to the score.

(9) `bias:` The biased composition score correction that was applied to the bit score.

(10) `#:` This domain's number (1..ndom).

(11) `of:` The total number of domains reported in the sequence, ndom.

(12) c-Evalue: The "conditional E-value", a permissive measure of how reliable this particular domain may be. The conditional E-value is calculated on a smaller search space than the independent E-value. The conditional E-value uses the number of targets that pass the reporting thresholds. The null hypothesis test posed by the conditional E-value is as follows. Suppose that we believe that there is already sufficient evidence (from other domains) to identify the set of reported sequences as homologs of our query; now, how many *additional* domains would we expect to find with at least this particular domain's bit score, if the rest of those reported sequences were random nonhomologous sequence (i.e. outside the other domain(s) that were sufficient to identified them as homologs in the first place)?

(13) i-Evalue: The "independent E-value", the E-value that the sequence/profile comparison would have received if this were the only domain envelope found in it, excluding any others. This is a stringent measure of how reliable this particular domain may be. The independent E-value uses the total number of targets in the target database.

(14) score: The bit score for this domain.

(15) bias: The biased composition (null2) score correction that was applied to the domain bit score.

(16) from (hmm coord): The start of the MEA alignment of this domain with respect to the profile, numbered 1..N for a profile of N consensus positions.

(17) to (hmm coord): The end of the MEA alignment of this domain with respect to the profile, numbered 1..N for a profile of N consensus positions.

(18) from (ali coord): The start of the MEA alignment of this domain with respect to the sequence, numbered 1..L for a sequence of L residues.

(19) to (ali coord): The end of the MEA alignment of this domain with respect to the sequence, numbered 1..L for a sequence of L residues.

(20) from (env coord): The start of the domain envelope on the sequence, numbered 1..L for a sequence of L residues. The *envelope* defines a subsequence for which their is substantial probability mass supporting a homologous domain, whether or not a single discrete alignment can be identified. The envelope may extend beyond the endpoints of the MEA alignment, and in fact often does, for weakly scoring domains.

(21) to (env coord): The end of the domain envelope on the sequence, numbered 1..L for a sequence of L residues.

(22) acc: The mean posterior probability of aligned residues in the MEA alignment; a measure of how reliable the overall alignment is (from 0 to 1, with 1.00 indicating a completely reliable alignment according to the model).

(23) description of target: The remainder of the line is the target's description line, as free text.

As with the target hits table (above), this table is columnated neatly for human readability, but you should not write parsers that rely on this columnation; parse based on space-delimited fields instead.

6 Some other topics

How do I cite HMMER?

The appropriate citation is to the web site, `hmmer.org`. You should also cite what version of the software you used. We archive all old versions, so anyone should be able to obtain the version you used, when exact reproducibility of an analysis is an issue.

The version number is in the header of most output files. To see it quickly, do something like `hmmscan -h` to get a help page, and the header will say:

```
# hmmscan :: search sequence(s) against a profile database
# HMMER 3.1 (February 2013); http://hmmer.org/
# Copyright (C) 2011 Howard Hughes Medical Institute.
# Freely distributed under the GNU General Public License (GPLv3).
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

So (from the second line there) this is from HMMER 3.1.

There is not yet any appropriate citable published paper that describes the HMMER3 software suite.

How do I report a bug?

Email us, at `hmmer@janelia.hhmi.org`.

Before we can see what needs fixing, we almost always need to reproduce a bug on one of our machines. This means we want to have a small, reproducible test case that shows us the failure you're seeing. So if you're reporting a bug, please send us:

- A brief description of what went wrong.

- The command line(s) that reproduce the problem.

- Copies of any files we need to run those command lines.

- Information about what kind of hardware you're on, what operating system, and (if you compiled the software yourself rather than running precompiled binaries), what compiler and version you used, with what configuration arguments.

Depending on how glaring the bug is, we may not need all this information, but any work you can put into giving us a clean reproducible test case doesn't hurt and often helps.

The information about hardware, operating system, and compiler is important. Bugs are frequently specific to particular configurations of hardware/OS/compiler. We have a wide variety of systems available for trying to reproduce bugs, and we'll try to match your system as closely as we can.

If you first see a problem on some huge compute (like running a zillion query sequence over a huge profile database), it will really, really help us if you spend a bit of time yourself trying to isolate whether the problem really only manifests itself on that huge compute, or if you can isolate a smaller test case for us. The ideal bug report (for us) gives us everything we need to reproduce your problem in one email with at most some small attachments.

Remember, we're not a company with dedicated support staff – we're a small lab of busy researchers like you. Somebody here is going to drop what they're doing to try to help you out. Try to save us some time, and we're more likely to stay in our usual good mood.

If we're in our usual good mood, we'll reply quickly. We'll probably tell you we fixed the bug in our development code, and that the fix will appear in the next HMMER release. This of course doesn't help you much, since nobody knows when the next HMMER release is going to be. So if possible, we'll usually try to describe a workaround for the bug.

If the code fix is small, we might also tell you how to patch and recompile the code yourself. You may or may not want to do this.

There are currently not enough open bugs to justify having a formal on-line bug tracking system. We have a bugtracking system, but it's internal.

Input files

Reading from a stdin pipe using - (dash) as a filename argument

Generally, HMMER programs read their sequence and/or profile input from files. Unix power users often find it convenient to string an incantation of commands together with pipes (indeed, such wizardly incantations are a point of pride). For example, you might extract a subset of query sequences from a larger file using a one-liner combination of scripting commands (perl, awk, whatever). To facilitate the use of HMMER programs in such incantations, you can almost always use an argument of '-' (dash) in place of a filename, and the program will take its input from a standard input pipe instead of opening a file.

For example, the following three commands are entirely equivalent, and give essentially identical output:

> **hmmsearch globins4.hmm uniprot_sprot.fasta**

> **cat globins4.hmm | hmmsearch − uniprot_sprot.fasta**

> **cat uniprot_sprot.fasta | hmmsearch globins4.hmm −**

Most Easel "miniapp" programs share the same ability of pipe-reading.

Because the programs for profile HMM fetching (`hmmfetch`) and sequence fetching (`esl-sfetch`) can fetch any number of profiles or sequences by names/accessions given in a list, *and* these programs can also read these lists from a stdin pipe, you can craft incantations that generate subsets of queries or targets on the fly. For example:

> **esl−sfetch −−index uniprot_sprot.fasta**

> **cat mytargs.list | esl−sfetch −f uniprot_sprot.fasta − | hmmsearch globins4.hmm −**

This takes a list of sequence names/accessions in `mytargets.list`, fetches them one by one from UniProt (note that we index the UniProt file first, for fast retrieval; and note that `esl-sfetch` is reading its `<namefile>` list of names/accessions through a pipe using the '-' argument), and pipes them to an `hmmsearch`. It should be obvious from this that we can replace the `cat mytargets.list` with *any* incantation that generates a list of sequence names/accessions (including SQL database queries).

Ditto for piping subsets of profiles. Supposing you have a copy of Pfam in Pfam-A.hmm:

> **hmmfetch −−index Pfam−A.hmm**

> **cat myqueries.list | hmmfetch −f Pfam.hmm − | hmmsearch − uniprot_sprot.fasta**

This takes a list of query profile names/accessions in `myqueries.list`, fetches them one by one from Pfam, and does an hmmsearch with each of them against UniProt. As above, the `cat myqueries.list` part can be replaced by any suitable incantation that generates a list of profile names/accessions.

There are three kinds of cases where using '-' is restricted or doesn't work. A fairly obvious restriction is that you can only use one '-' per command; you can't do a `hmmsearch - -` that tries to read both profile queries and sequence targets through the same stdin pipe. Second, another case is when an input file must be obligately associated with additional, separately generated auxiliary files, so reading data from a single stream using '-' doesn't work because the auxiliary files aren't present (in this case, using '-' will be prohibited by the program). An example is `hmmscan`, which needs its `<hmmfile>` argument to be associated with four auxiliary files named `<hmmfile>`.h3{mifp} that `hmmpress` creates, so `hmmscan` does not permit a '-' for its `<hmmfile>` argument. Finally, when a command would require multiple passes over an input file, the command will generally abort after the first pass if you are trying to read that file through a standard input pipe (pipes are nonrewindable in general; a few HMMER or Easel programs will buffer input streams to make multiple passes possible, but this is not usually the case). An example would be trying to search a file containing multiple profile queries against a streamed target database:

> **cat myqueries.list | hmmfetch −f Pfam.hmm > many.hmms**

> **cat mytargets.list | esl−sfetch −f uniprot_sprot.fasta − | hmmsearch many.hmms −**

This will fail. Unfortunately the above business about how it will "generally abort after the first pass" means it fails weirdly. The first query profile search will succeed, and its output will appear; then an error message will be generated when `hmmsearch` sees the *second* profile query and oops, it realizes it is unable to rewind the target sequence database stream. This is inherent in how it reads the profile HMM query file sequentially as a stream (which is what's allowing it to read input from stdin pipes in the first place), one

model at a time: it doesn't see there's more than one query model in the file until it gets to the second model.

This case isn't too restricting because the same end goal can be achieved by reordering the commands. In cases where you want to do multiple queries against multiple targets, you always want to be reading the *queries* from a stdin pipe, not the targets:

```
> cat mytargets.list | esl-sfetch -f uniprot_sprot.fasta > mytarget.seqs
```

```
> cat myqueries.list | hmmfetch -f Pfam.hmm - | hmmsearch - mytarget.seqs
```

So in this multiple queries/multiple targets case of using stdin pipes, you just have to know, for any given program, which file it considers to be queries and which it considers to be targets. (That is, the logic in searching many queries against many targets is "For each query: search the target database; then rewind the target database to the beginning.") For `hmmsearch`, the profiles are queries and sequences are targets. For `hmmscan`, the reverse.

In general, HMMER and Easel programs document in their man page whether (and which) command line arguments can be replaced by '-'. You can always check by trial and error, too. The worst that can happen is a "Failed to open file -" error message, if the program can't read from pipes.

7 Manual pages

`alimask` - Add mask line to a multiple sequence alignment

Synopsis

alimask *[options]* <*msafile*> <*postmsafile*>

Description

alimask is used to apply a mask line to a multiple sequence alignment, based on provided alignment or model coordinates. When **hmmbuild** receives a masked alignment as input, it produces a profile model in which the emission probabilities at masked positions are set to match the background frequency, rather than being set based on observed frequencies in the alignment. Position-specific insertion and deletion rates are not altered, even in masked regions. **alimask** autodetects input format, and produces masked alignments in Stockholm format. <*msafile*> may contain only one sequence alignment.

A common motivation for masking a region in an alignment is that the region contains a simple tandem repeat that is observed to cause an unacceptably high rate of false positive hits.

In the simplest case, a mask range is given in coordinates relative to the input alignment, using **--alirange** <*s*>. However it is more often the case that the region to be masked has been identified in coordinates relative to the profile model (e.g. based on recognizing a simple repeat pattern in false hit alignments or in the HMM logo). Not all alignment columns are converted to match state positions in the profile (see the *--symfrac* flag for **hmmbuild** for discussion), so model positions do not necessarily match up to alignment column positions. To remove the burden of converting model positions to alignment positions, **alimask** accepts the mask range input in model coordinates as well, using **--modelrange** <*s*>. When using this flag, *alimask* determines which alignment positions would be identified by **hmmbuild** as match states, a process that requires that all **hmmbuild** flags impacting that decision be supplied to **alimask**. It is for this reason that many of the **hmmbuild** flags are also used by **alimask**.

Options

> **-h** Help; print a brief reminder of command line usage and all available options.

> **-o** <*f*> Direct the summary output to file <*f*>, rather than to *stdout*.

Options for Specifying Mask Range

A single mask range is given as a dash-separated pair, like *--modelrange 10-20* and multiple ranges may be submitted as a comma-separated list, *--modelrange 10-20,30-42*.

> **--modelrange** <*s*> Supply the given range(s) in model coordinates.

> **--alirange** <*s*> Supply the given range(s) in alignment coordinates.

> **--apendmask** Add to the existing mask found with the alignment. The default is to overwrite any existing mask.

--model2ali $<s>$ Rather than actually produce the masked alignment, simply print model range(s) corresponding to input alignment range(s).

--ali2model $<s>$ Rather than actually produce the masked alignment, simply print alignment range(s) corresponding to input model range(s).

Options for Specifying the Alphabet

The alphabet type (amino, DNA, or RNA) is autodetected by default, by looking at the composition of the *msafile*. Autodetection is normally quite reliable, but occasionally alphabet type may be ambiguous and autodetection can fail (for instance, on tiny toy alignments of just a few residues). To avoid this, or to increase robustness in automated analysis pipelines, you may specify the alphabet type of *msafile* with these options.

--amino Specify that all sequences in *msafile* are proteins.

--dna Specify that all sequences in *msafile* are DNAs.

--rna Specify that all sequences in *msafile* are RNAs.

Options Controlling Profile Construction

These options control how consensus columns are defined in an alignment.

--fast Define consensus columns as those that have a fraction $>=$ **symfrac** of residues as opposed to gaps. (See below for the **--symfrac** option.) This is the default.

--hand Define consensus columns in next profile using reference annotation to the multiple alignment. This allows you to define any consensus columns you like.

--symfrac $<x>$ Define the residue fraction threshold necessary to define a consensus column when using the **--fast** option. The default is 0.5. The symbol fraction in each column is calculated after taking relative sequence weighting into account, and ignoring gap characters corresponding to ends of sequence fragments (as opposed to internal insertions/deletions). Setting this to 0.0 means that every alignment column will be assigned as consensus, which may be useful in some cases. Setting it to 1.0 means that only columns that include 0 gaps (internal insertions/deletions) will be assigned as consensus.

--fragthresh $<x>$ We only want to count terminal gaps as deletions if the aligned sequence is known to be full-length, not if it is a fragment (for instance, because only part of it was sequenced). HMMER uses a simple rule to infer fragments: if the sequence length L is less than or equal to a fraction $<x>$ times the alignment length in columns, then the sequence is handled as a fragment. The default is 0.5. Setting **--fragthresh***0* will define no (nonempty) sequence as a fragment; you might want to do this if you know you've got a carefully curated alignment of full-length sequences. Setting **--fragthresh***1* will define all sequences as fragments; you might want to do this if you know your alignment is entirely composed of fragments, such as translated short reads in metagenomic shotgun data.

Options Controlling Relative Weights

HMMER uses an ad hoc sequence weighting algorithm to downweight closely related sequences and up-weight distantly related ones. This has the effect of making models less biased by uneven phylogenetic representation. For example, two identical sequences would typically each receive half the weight that one sequence would. These options control which algorithm gets used.

--wpb Use the Henikoff position-based sequence weighting scheme [Henikoff and Henikoff, J. Mol. Biol. 243:574, 1994]. This is the default.

--wgsc Use the Gerstein/Sonnhammer/Chothia weighting algorithm [Gerstein et al, J. Mol. Biol. 235:1067, 1994].

--wblosum Use the same clustering scheme that was used to weight data in calculating BLO-SUM subsitution matrices [Henikoff and Henikoff, Proc. Natl. Acad. Sci 89:10915, 1992]. Sequences are single-linkage clustered at an identity threshold (default 0.62; see --*wid*) and within each cluster of c sequences, each sequence gets relative weight 1/c.

--wnone No relative weights. All sequences are assigned uniform weight.

--wid $<x>$ Sets the identity threshold used by single-linkage clustering when using --*wblosum*. Invalid with any other weighting scheme. Default is 0.62.

Other Options

--informat $<s>$ Declare that the input *msafile* is in format $<s>$. Currently the accepted multiple alignment sequence file formats include Stockholm, Aligned FASTA, Clustal, NCBI PSI-BLAST, PHYLIP, Selex, and UCSC SAM A2M. Default is to autodetect the format of the file.

--seed $<n>$ Seed the random number generator with $<n>$, an integer $>=$ 0. If $<n>$ is nonzero, any stochastic simulations will be reproducible; the same command will give the same results. If $<n>$ is 0, the random number generator is seeded arbitrarily, and stochastic simulations will vary from run to run of the same command. The default seed is 42.

`hmmalign` - align sequences to a profile HMM

Synopsis

hmmalign *[options]* *<hmmfile> <seqfile>*

Description

Perform a multiple sequence alignment of all the sequences in *<seqfile>* by aligning them individually to the profile HMM in *<hmmfile>*. The new alignment is output to *stdout* in Stockholm format.

The *<hmmfile>* should contain only a single profile. If it contains more, only the first profile in the file will be used.

Either *<hmmfile>* or *<seqfile>* (but not both) may be '-' (dash), which means reading this input from *stdin* rather than a file.

The sequences in *<seqfile>* are aligned in unihit local alignment mode. Therefore they should already be known to contain only a single domain (or a fragment of one). The optimal alignment may assign some residues as nonhomologous (N and C states), in which case these residues are still included in the resulting alignment, but shoved to the outer edges. To trim these unaligned nonhomologous residues from the result, see the **--trim** option.

Options

-h	Help; print a brief reminder of command line usage and all available options.
-o *<f>*	Direct the output alignment to file *<f>*, rather than to *stdout.*
--mapali *<f>*	Merge the existing alignment in file *<f>* into the result, where *<f>* is exactly the same alignment that was used to build the model in *<hmmfile>*. This is done using a map of alignment columns to consensus profile positions that is stored in the *<hmmfile>*. The multiple alignment in *<f>* will be exactly reproduced in its consensus columns (as defined by the profile), but the displayed alignment in insert columns may be altered, because insertions relative to a profile are considered by convention to be unaligned data.
--trim	Trim nonhomologous residues (assigned to N and C states in the optimal alignments) from the resulting multiple alignment output.
--amino	Specify that all sequences in *<seqfile>* are proteins. By default, alphabet type is autodetected from looking at the residue composition.
--dna	Specify that all sequences in *<seqfile>* are DNAs.
--rna	Specify that all sequences in *<seqfile>* are RNAs.
--informat *<s>*	Declare that the input *<seqfile>* is in format *<s>*. Accepted sequence file formats include FASTA, EMBL, GenBank, DDBJ, UniProt, Stockholm, and SELEX. Default is to autodetect the format of the file.
--outformat *<s>*	Specify that the output multiple alignment is in format *<s>*. Currently the accepted multiple alignment sequence file formats only include Stockholm and SELEX.

hmmbuild - construct profile HMM(s) from multiple sequence alignment(s)

Synopsis

hmmbuild *[options]* *<hmmfile_out>* *<msafile>*

Description

For each multiple sequence alignment in *<msafile>* build a profile HMM and save it to a new file *<hmmfile_out>*.

<msafile> may be '-' (dash), which means reading this input from *stdin* rather than a file. To use '-', you must also specify the alignment file format with **--informat** *<s>*, as in **--informat stockholm** (because of a current limitation in our implementation, MSA file formats cannot be autodetected in a nonrewindable input stream.)

<hmmfile_out> may not be '-' *(stdout)*, because sending the HMM file to *stdout* would conflict with the other text output of the program.

Options

- **-h** Help; print a brief reminder of command line usage and all available options.

- **-n** *<s>* Name the new profile *<s>*. The default is to use the name of the alignment (if one is present in the *msafile*, or, failing that, the name of the *hmmfile*. If *msafile* contains more than one alignment, *-n* doesn't work, and every alignment must have a name annotated in the *msafile* (as in Stockholm #=GF ID annotation).

- **-o** *<f>* Direct the summary output to file *<f>*, rather than to *stdout*.

- **-O** *<f>* After each model is constructed, resave annotated, possibly modified source alignments to a file *<f>* in Stockholm format. The alignments are annotated with a reference annotation line indicating which columns were assigned as consensus, and sequences are annotated with what relative sequence weights were assigned. Some residues of the alignment may have been shifted to accommodate restrictions of the Plan7 profile architecture, which disallows transitions between insert and delete states.

Options for Specifying the Alphabet

The alphabet type (amino, DNA, or RNA) is autodetected by default, by looking at the composition of the *msafile*. Autodetection is normally quite reliable, but occasionally alphabet type may be ambiguous and autodetection can fail (for instance, on tiny toy alignments of just a few residues). To avoid this, or to increase robustness in automated analysis pipelines, you may specify the alphabet type of *msafile* with these options.

- **--amino** Specify that all sequences in *msafile* are proteins.

- **--dna** Specify that all sequences in *msafile* are DNAs.

- **--rna** Specify that all sequences in *msafile* are RNAs.

Options Controlling Profile Construction

These options control how consensus columns are defined in an alignment.

--fast   Define consensus columns as those that have a fraction $>=$ **symfrac** of residues as opposed to gaps. (See below for the **--symfrac** option.) This is the default.

--hand   Define consensus columns in next profile using reference annotation to the multiple alignment. This allows you to define any consensus columns you like.

--symfrac $<x>$ Define the residue fraction threshold necessary to define a consensus column when using the **--fast** option. The default is 0.5. The symbol fraction in each column is calculated after taking relative sequence weighting into account, and ignoring gap characters corresponding to ends of sequence fragments (as opposed to internal insertions/deletions). Setting this to 0.0 means that every alignment column will be assigned as consensus, which may be useful in some cases. Setting it to 1.0 means that only columns that include 0 gaps (internal insertions/deletions) will be assigned as consensus.

--fragthresh $<x>$ We only want to count terminal gaps as deletions if the aligned sequence is known to be full-length, not if it is a fragment (for instance, because only part of it was sequenced). HMMER uses a simple rule to infer fragments: if the range of a sequence in the alignment (the number of alignment columns between the first and last positions of the sequence) is less than or equal to a fraction $<x>$ times the alignment length in columns, then the sequence is handled as a fragment. The default is 0.5. Setting **--fragthresh**0 will define no (nonempty) sequence as a fragment; you might want to do this if you know you've got a carefully curated alignment of full-length sequences. Setting **--fragthresh**1 will define all sequences as fragments; you might want to do this if you know your alignment is entirely composed of fragments, such as translated short reads in metagenomic shotgun data.

Options Controlling Relative Weights

HMMER uses an ad hoc sequence weighting algorithm to downweight closely related sequences and up-weight distantly related ones. This has the effect of making models less biased by uneven phylogenetic representation. For example, two identical sequences would typically each receive half the weight that one sequence would. These options control which algorithm gets used.

--wpb   Use the Henikoff position-based sequence weighting scheme [Henikoff and Henikoff, J. Mol. Biol. 243:574, 1994]. This is the default.

--wgsc   Use the Gerstein/Sonnhammer/Chothia weighting algorithm [Gerstein et al, J. Mol. Biol. 235:1067, 1994].

--wblosum   Use the same clustering scheme that was used to weight data in calculating BLO-SUM subsitution matrices [Henikoff and Henikoff, Proc. Natl. Acad. Sci 89:10915, 1992]. Sequences are single-linkage clustered at an identity threshold (default 0.62; see *--wid*) and within each cluster of c sequences, each sequence gets relative weight 1/c.

--wnone   No relative weights. All sequences are assigned uniform weight.

--wid $<x>$ Sets the identity threshold used by single-linkage clustering when using *--wblosum*. Invalid with any other weighting scheme. Default is 0.62.

Options Controlling Effective Sequence Number

After relative weights are determined, they are normalized to sum to a total effective sequence number, *eff_nseq*. This number may be the actual number of sequences in the alignment, but it is almost always smaller than that. The default entropy weighting method *(--eent)* reduces the effective sequence number to reduce the information content (relative entropy, or average expected score on true homologs) per consensus position. The target relative entropy is controlled by a two-parameter function, where the two parameters are settable with *--ere* and *--esigma*.

--eent   Adjust effective sequence number to achieve a specific relative entropy per position (see *--ere*). This is the default.

--eclust   Set effective sequence number to the number of single-linkage clusters at a specific identity threshold (see *--eid*). This option is not recommended; it's for experiments evaluating how much better **--eent** is.

--enone   Turn off effective sequence number determination and just use the actual number of sequences. One reason you might want to do this is to try to maximize the relative entropy/position of your model, which may be useful for short models.

--eset $<x>$ Explicitly set the effective sequence number for all models to $<x>$.

--ere $<x>$ Set the minimum relative entropy/position target to $<x>$. Requires **--eent**. Default depends on the sequence alphabet. For protein sequences, it is 0.59 bits/position; for nucleotide sequences, it is 0.45 bits/position.

--esigma $<x>$ Sets the minimum relative entropy contributed by an entire model alignment, over its whole length. This has the effect of making short models have higher relative entropy per position than *--ere* alone would give. The default is 45.0 bits.

--eid $<x>$ Sets the fractional pairwise identity cutoff used by single linkage clustering with the **--eclust** option. The default is 0.62.

Options Controlling Priors

By default, weighted counts are converted to mean posterior probability parameter estimates using mixture Dirichlet priors. Default mixture Dirichlet prior parameters for protein models and for nucleic acid (RNA and DNA) models are built in. The following options allow you to override the default priors.

--pnone   Don't use any priors. Probability parameters will simply be the observed frequencies, after relative sequence weighting.

--plaplace   Use a Laplace +1 prior in place of the default mixture Dirichlet prior.

Options Controlling E-value Calibration

The location parameters for the expected score distributions for MSV filter scores, Viterbi filter scores, and Forward scores require three short random sequence simulations.

--EmL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EmN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EvL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EvN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EfL $<n>$ Sets the sequence length in simulation that estimates the location parameter tau for Forward E-values. Default is 100.

--EfN $<n>$ Sets the number of sequences in simulation that estimates the location parameter tau for Forward E-values. Default is 200.

--Eft $<x>$ Sets the tail mass fraction to fit in the simulation that estimates the location parameter tau for Forward evalues. Default is 0.04.

Other Options

--cpu $<n>$ Set the number of parallel worker threads to $<n>$. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting $<n>$ higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off for your site or machine for some reason.

--informat $<s>$ Declare that the input *msafile* is in format $<s>$. Currently the accepted multiple alignment sequence file formats include Stockholm, Aligned FASTA, Clustal, NCBI PSI-BLAST, PHYLIP, Selex, and UCSC SAM A2M. Default is to autodetect the format of the file.

--seed $<n>$ Seed the random number generator with $<n>$, an integer $>= 0$. If $<n>$ is nonzero, any stochastic simulations will be reproducible; the same command will give the same results. If $<n>$ is 0, the random number generator is seeded arbitrarily, and stochastic simulations will vary from run to run of the same command. The default seed is 42.

--w_beta $<x>$ Window length tail mass. The upper bound, *W*, on the length at which nhmmer expects to find an instance of the model is set such that the fraction of all sequences generated by the model with length $>= W$ is less than $<x>$. The default is 1e-7.

--w_length $<n>$ Override the model instance length upper bound, W, which is otherwise controlled by **--w_beta**. It should be larger than the model length. The value of W is used deep in the acceleration pipeline, and modest changes are not expected to impact results (though larger values of W do lead to longer run time).

--mpi   Run as a parallel MPI program. Each alignment is assigned to a MPI worker node for construction. (Therefore, the maximum parallelization cannot exceed the number of alignments in the input *msafile*.) This is useful when building large profile libraries. This option is only available if optional MPI capability was enabled at compile-time.

--stall   For debugging MPI parallelization: arrest program execution immediately after start, and wait for a debugger to attach to the running process and release the arrest.

--maxinsertlen $<n>$ Restrict insert length parameterization such that the expected insert length at each position of the model is no more than $<n>$.

`hmmconvert` - convert profile file to a HMMER format

Synopsis

hmmconvert *[options]* <*hmmfile*>

Description

The **hmmconvert** utility converts an input profile file to different HMMER formats.

By default, the input profile can be in any HMMER format, including old/obsolete formats from HMMER2, ASCII or binary; the output profile is a current HMMER3 ASCII format.

<*hmmfile*> may be '-' (dash), which means reading this input from *stdin* rather than a file.

Options

    -h Help; print a brief reminder of command line usage and all available options.

    -a Output profiles in ASCII text format. This is the default.

    -b Output profiles in binary format.

    -2 Output in legacy HMMER2 ASCII text format, in ls (glocal) mode. This allows HMMER3 models to be converted back to a close approximation of HMMER2, for comparative studies.

    --outfmt <*s*> Output in a HMMER3 ASCII text format other then the most current one. Valid choices for <*s*> are 3/a through 3/f. The current format is 3/f, and this is the default. The format 3/b was used in the official HMMER3 release, and the others were used in the various testing versions.

`hmmemit` - sample sequences from a profile HMM

Synopsis

hmmemit *[options]* *hmmfile*

Description

The **hmmemit** program samples (emits) sequences from the profile HMM(s) in *hmmfile*, and writes them to output. Sampling sequences may be useful for a variety of purposes, including creating synthetic true positives for benchmarks or tests.

The default is to sample one unaligned sequence from the core probability model, which means that each sequence consists of one full-length domain. Alternatively, with the **-c** option, you can emit a simple majority-rule consensus sequence; or with the **-a** option, you can emit an alignment (in which case, you probably also want to set **-N** to something other than its default of 1 sequence per model).

As another option, with the **-p** option you can sample a sequence from a fully configured HMMER search profile. This means sampling a 'homologous sequence' by HMMER's definition, including nonhomologous flanking sequences, local alignments, and multiple domains per sequence, depending on the length model and alignment mode chosen for the profile.

The *hmmfile* may contain a library of HMMs, in which case each HMM will be used in turn.

<*hmmfile*> may be '-' (dash), which means reading this input from *stdin* rather than a file.

Common Options

> **-h** Help; print a brief reminder of command line usage and all available options.
>
> **-o** <*f*> Direct the output sequences to file <*f*>, rather than to *stdout*.
>
> **-N** <*n*> Sample <*n*> sequences per model, rather than just one.

Options Controlling What to Emit

The default is to sample **N** sequences from the core model. Alternatively, you may choose one (and only one) of the following alternatives.

> **-a** Emit an alignment for each HMM in the *hmmfile* rather than sampling unaligned sequences one at a time.
>
> **-c** Emit a plurality-rule consensus sequence, instead of sampling a sequence from the profile HMM's probability distribution. The consensus sequence is formed by selecting the maximum probability residue at each match state.
>
> **-C** Emit a fancier plurality-rule consensus sequence than the **-c** option. If the maximum probability residue has $p <$ **minl** show it as a lower case 'any' residue (n or x); if $p >=$ **minl** and $<$ **minu** show it as a lower case residue; and if $p >=$ **minu** show it as an upper case residue. The default settings of **minu** and **minl** are both 0.0, which means **-C** gives the same output as **-c** unless you also set **minu** and **minl** to what you want.

-p   Sample unaligned sequences from the implicit search profile, not from the core model. The core model consists only of the homologous states (between the begin and end states of a HMMER Plan7 model). The profile includes the nonhomologous N, C, and J states, local/glocal and uni/multihit algorithm configuration, and the target length model. Therefore sequences sampled from a profile may include nonhomologous as well as homologous sequences, and may contain more than one homologous sequence segment. By default, the profile is in multihit local mode, and the target sequence length is configured for L=400.

Options Controlling Emission from Profiles

These options require that you have set the **-p** option.

-L $<n>$ Configure the profile's target sequence length model to generate a mean length of approximately $<n>$ rather than the default of 400.

--local   Configure the profile for multihit local alignment.

--unilocal   Configure the profile for unihit local alignment (Smith/Waterman).

--glocal   Configure the profile for multihit glocal alignment.

--uniglocal   Configure the profile for unihit glocal alignment.

Options Controlling Fancy Consensus Emission

These options require that you have set the **-C option**.

--minl $<x>$ Sets the **minl** threshold for showing weakly conserved residues as lower case. ($0 <= x <= 1$)

--minu $<x>$ Sets the **minu** threshold for showing strongly conserved residues as upper case. ($0 <= x <= 1$)

Other Options

--seed $<n>$ Seed the random number generator with $<n>$, an integer $>= 0$. If $<n>$ is nonzero, any stochastic simulations will be reproducible; the same command will give the same results. If $<n>$ is 0, the random number generator is seeded arbitrarily, and stochastic simulations will vary from run to run of the same command. The default is 0: use an arbitrary seed, so different **hmmemit** runs will generate different samples.

`hmmfetch` - retrieve profile HMM(s) from a file

Synopsis

hmmfetch *[options]* <*hmmfile*> <*key*> (retrieves HMM named <key>)

hmmfetch -f *[options]* <*hmmfile*> <*keyfile*> (retrieves all HMMs listed in <keyfile>)

hmmfetch --index *[options]* <*hmmfile*> (indexes <hmmfile> for fetching)

Description

Quickly retrieves one or more profile HMMs from an <*hmmfile*> (a large Pfam database, for example).

For maximum speed, the <*hmmfile*> should be indexed first, using **hmmfetch --index**. The index is a binary file named <*hmmfile*>*.ssi*. However, this is optional, and retrieval will still work from unindexed files, albeit much more slowly.

The default mode is to retrieve a single profile by name or accession, called the <*key*>. For example:

% hmmfetch Pfam-A.hmm Caudal_act

% hmmfetch Pfam-A.hmm PF00045

With the *-f* option, a <*keyfile*> containing a list of one or more keys is read instead. The first whitespace-delimited field on each non-blank non-comment line of the <*keyfile*> is used as a <*key*>, and any remaining data on the line is ignored. This allows a variety of whitespace delimited datafiles to be used as <*keyfile*>s.

When using *-f* and a <*keyfile*>, if **hmmfile** has been indexed, the keys are retrieved in the order they occur in the *keyfile*, but if **hmmfile** isn't indexed, keys are retrieved in the order they occur in the **hmmfile**. This is a side effect of an implementation that allows multiple keys to be retrieved even if the <**hmmfile**> is a nonrewindable stream, like a standard input pipe.

In normal use (without *--index* or *-f* options), <*hmmfile*> may be '-' (dash), which means reading input from *stdin* rather than a file. With the *--index* option, <*hmmfile*> may not be '-'; it does not make sense to index a standard input stream. With the *-f* option, either <*hmmfile*> or <*keyfile*> (but not both) may be '-'. It is often particularly useful to read <*keyfile*> from standard input, because this allows use to use arbitrary command line invocations to create a list of HMM names or accessions, then fetch them all to a new file, just with one command.

By default, fetched HMMs are printed to standard output in HMMER3 format.

Options

-h Help; print a brief reminder of command line usage and all available options.

-f The second commandline argument is a <*keyfile*> instead of a single <*key*>. The first field on each line of the <*keyfile*> is used as a retrieval <*key*> (an HMM name or accession). Blank lines and comment lines (that start with a # character) are ignored.

-o <*f*> Output HMM(s) to file <*f*> instead of to standard output.

-O Output HMM(s) to individual file(s) named <*key*> instead of standard output. With the **-f** option, this can result in many files being created.

--index Instead of retrieving one or more profiles from *<hmmfile>*, index the *<hmmfile>* for future retrievals. This creates a *<hmmfile>.ssi* binary index file.

`hmmlogo` - given an HMM, produce data required to build an HMM logo

Synopsis

hmmlogo *[options]* *<hmmfile>*

Description

hmmlogo computes letter height and indel parameters that can be used to produce a profile HMM logo. This tool is essentially a command-line interface for much of the data underlying the Skylign logo server (skylign.org). By default, **hmmlogo** prints out a table of per-position letter heights (dependent on the requested height method), then prints a table of per-position gap probabilities. In a typical logo, the total height of a stack of letters for one position depends on the information content of the position, and that stack height is subdivided according to the emission probabilities of the letters of the alphabet.

Options

-h	Help; print a brief reminder of command line usage and all available options.
--height_relent_all	Total height = relative entropy (aka information content); all letters are given a positive height. (default)
--height_relent_abovebg	Total height = relative entropy (aka information content); only letters with above-background probability are given positive height.
--height_score	Total height = sums of scores of positive-scoring letters; letter height depends on the score of that letter at that position. Only letters with above-background probability (positive score) are given positive height. (Note that only letter height is meaningful - stack height has no inherent meaning).
--no_indel	Don't print out the indel probability table.

hmmpgmd - daemon for searching a protein query against a protein database

Synopsis

hmmpgmd *[options]*

Description

The **hmmpgmd** program is the daemon that we use internally for the hmmer.org web server, and essentially stands in front of the protein search programs **phmmer**, **hmmsearch**, and **hmmscan**. To use **hmmpgmd**, first an instance must be started up as a *master* server, and provided with at least one of a *sequence database* (using the **--seqdb** flag) and/or an *HMM database* (using the **--hmmdb** flag). A sequence database must be in the hmmpgmd format, which may be produced using **esl-reformat**. An HMM database is of the form produced by **hmmbuild**. The input database(s) will be loaded into memory by the master. When the master has finished loading the database(s), it prints the line: "Data loaded into memory. Master is ready."

Only after master is ready, one or more instances of hmmpgmd may be started as workers. These workers may be (and typically are) on different machines from the master, but must have access to the same database file(s) provided to the master, with the same path. As with the master, each worker loads the database(s) into memory, and indicates completion by printing: "Data loaded into memory. Worker is ready."

The master server and workers are expected to remain running. One or more clients then connect to the master and submit possibly many queries. The master distributes the work of a query among the workers, collects results, and merges them before responding to the client. Two example client programs are included in the HMMER3.1 src directory - the C program **hmmc2** and the perl script **hmmpgmd_client_example.pl**. These are intended as examples only, and should be extended as necessary to meet your needs.

A query is submitted to the master from the client as a character string. Queries may be the sort that would normally be handled by **phmmer** (protein sequence vs protein database), **hmmsearch** (protein HMM query vs protein database), or **hmmscan** (protein query vs protein HMM database).

The general form of a client query is to start with a single line of the form *@[options]*, followed by multiple lines of text representing either the query HMM or fasta-formatted sequence. The final line of each query is the separator *//*.

For example, to perform a **phmmer** type search of a sequence against a sequence database file, the first line is of the form *@--seqdb 1*, then the fasta-formatted query sequence starting with the header line *>sequence-name*, followed by one or more lines of sequence, and finally the closing *//*.

To perform an **hmmsearch** type search, the query sequence is replaced by the full text of a HMMER-format query HMM.

To perform an **hmmscan** type search, the text matches that of the **phmmer** type search, except that the first line changes to *@--hmmdb 1*.

In the hmmpgmd-formatted sequence database file, each sequence can be associated with one or more sub-databases. The *--seqdb* flag indicates which of these sub-databases will be queried. The HMM database format does not support sub-databases.

The result of each query is an undocumented data structure in binary format. In the future the data will be returned in a proper serialized structure, but for now, it requires meticulous unpacking within the client. The example clients show how this is done.

Options

      -h   Help; print a brief reminder of command line usage and all available options.

Expert Options

      --master   Run as the master server.

  --worker *<s>*   Run as a worker, connecting to the master server that is running on IP address *<s>*.

    --daemon   Run as a daemon using config file: /etc/hmmpgmd.conf

  --cport *<n>*   Port to use for communication between clients and the master server. The default is 51371.

  --wport *<n>*   Port to use for communication between workers and the master server. The default is 51372.

  --ccncts *<n>*   Maximum number of client connections to accept. The default is 16.

  --wcncts *<n>*   Maximum number of worker connections to accept. The default is 32.

    --pid *<f>*   Name of file into which the process id will be written.

  --seqdb *<f>*   Name of the file (in hmmpgmd format) containing protein sequences. The contents of this file will be cached for searches.

  --hmmdb *<f>*   Name of the file containing protein HMMs. The contents of this file will be cached for searches.

    --cpu *<n>*   Number of parallel threads to use (for *--worker*).

`hmmpress` - prepare an HMM database for hmmscan

Synopsis

hmmpress *[options]* *<hmmfile>*

Description

Starting from a profile database *<hmmfile>* in standard HMMER3 format, construct binary compressed datafiles for **hmmscan**. The *hmmpress* step is required for **hmmscan** to work.

Four files are created: *<hmmfile>.h3m*, *<hmmfile>.h3i*, *<hmmfile>.h3f*, and *<hmmfile>.h3p*. The *<hmmfile>.h3m* file contains the profile HMMs and their annotation in a binary format. The *<hmmfile>.h3i* file is an SSI index for the *<hmmfile>.h3m* file. The *<hmmfile>.h3f* file contains precomputed data structures for the fast heuristic filter (the MSV filter). The *<hmmfile>.h3p* file contains precomputed data structures for the rest of each profile.

<hmmfile> may not be '-' (dash); running **hmmpress** on a standard input stream rather than a file is not allowed.

Options

    -h Help; print a brief reminder of command line usage and all available options.

    -f Force; overwrites any previous hmmpress'ed datafiles. The default is to bitch about any existing files and ask you to delete them first.

`hmmscan` - search protein sequence(s) against a protein profile database

Synopsis

hmmscan *[options]* *<hmmdb>* *<seqfile>*

Description

hmmscan is used to search protein sequences against collections of protein profiles. For each sequence in *<seqfile>*, use that query sequence to search the target database of profiles in *<hmmdb>*, and output ranked lists of the profiles with the most significant matches to the sequence.

The *<seqfile>* may contain more than one query sequence. It can be in FASTA format, or several other common sequence file formats (genbank, embl, and uniprot, among others), or in alignment file formats (stockholm, aligned fasta, and others). See the *--qformat* option for a complete list.

The *<hmmdb>* needs to be press'ed using **hmmpress** before it can be searched with **hmmscan**. This creates four binary files, suffixed .h3{fimp}.

The query *<seqfile>* may be '-' (a dash character), in which case the query sequences are read from a *<stdin>* pipe instead of from a file. The *<hmmdb>* cannot be read from a *<stdin>* stream, because it needs to have those four auxiliary binary files generated by **hmmpress**.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** and **--domtblout** options save output in simple tabular formats that are concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

>   **-h** Help; print a brief reminder of command line usage and all available options.

Options for Controlling Output

>   **-o** *<f>* Direct the main human-readable output to a file *<f>* instead of the default stdout.

>   **--tblout** *<f>* Save a simple tabular (space-delimited) file summarizing the per-target output, with one data line per homologous target model found.

> **--domtblout** *<f>* Save a simple tabular (space-delimited) file summarizing the per-domain output, with one data line per homologous domain detected in a query sequence for each homologous model.

> **--pfamtblout** *<f>* Save an especially succinct tabular (space-delimited) file summarizing the per-target output, with one data line per homologous target model found.

>   **--acc** Use accessions instead of names in the main output, where available for profiles and/or sequences.

>   **--noali** Omit the alignment section from the main output. This can greatly reduce the output volume.

--notextw Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

--textw $<n>$ Set the main output's line length limit to $<n>$ characters per line. The default is 120.

Options for Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--domtblout*).

-E $<x>$ In the per-target output, report target profiles with an E-value of $<= <x>$. The default is 10.0, meaning that on average, about 10 false positives will be reported per query, so you can see the top of the noise and decide for yourself if it's really noise.

-T $<x>$ Instead of thresholding per-profile output on E-value, instead report target profiles with a bit score of $>= <x>$.

--domE $<x>$ In the per-domain output, for target profiles that have already satisfied the per-profile reporting threshold, report individual domains with a conditional E-value of $<= <x>$. The default is 10.0. A conditional E-value means the expected number of additional false positive domains in the smaller search space of those comparisons that already satisfied the per-profile reporting threshold (and thus must have at least one homologous domain already).

--domT $<x>$ Instead of thresholding per-domain output on E-value, instead report domains with a bit score of $>= <x>$.

Options for Inclusion Thresholds

Inclusion thresholds are stricter than reporting thresholds. Inclusion thresholds control which hits are considered to be reliable enough to be included in an output alignment or a subsequent search round. In **hmmscan**, which does not have any alignment output (like **hmmsearch** or **phmmer**) nor any iterative search steps (like **jackhmmer**), inclusion thresholds have little effect. They only affect what domains get marked as significant (!) or questionable (?) in domain output.

--incE $<x>$ Use an E-value of $<= <x>$ as the per-target inclusion threshold. The default is 0.01, meaning that on average, about 1 false positive would be expected in every 100 searches with different query sequences.

--incT $<x>$ Instead of using E-values for setting the inclusion threshold, instead use a bit score of $>= <x>$ as the per-target inclusion threshold. It would be unusual to use bit score thresholds with *hmmscan*, because you don't expect a single score threshold to work for different profiles; different profiles have slightly different expected score distributions.

--incdomE $<x>$ Use a conditional E-value of $<= <x>$ as the per-domain inclusion threshold, in targets that have already satisfied the overall per-target inclusion threshold. The default is 0.01.

--incdomT $<x>$ Instead of using E-values, instead use a bit score of $>= <x>$ as the per-domain inclusion threshold. As with **--incT** above, it would be unusual to use a single bit score threshold in **hmmscan**.

Options for Model-specific Score Thresholding

Curated profile databases may define specific bit score thresholds for each profile, superseding any thresholding based on statistical significance alone. To use these options, the profile must contain the appropriate (GA, TC, and/or NC) optional score threshold annotation; this is picked up by **hmmbuild** from Stockholm format alignment files. Each thresholding option has two scores: the per-sequence threshold $<x1>$ and the per-domain threshold $<x2>$ These act as if **-T**$<x1>$ **--incT**$<x1>$ **--domT**$<x2>$ **--incdomT**$<x2>$ has been applied specifically using each model's curated thresholds.

--cut_ga Use the GA (gathering) bit scores in the model to set per-sequence (GA1) and per-domain (GA2) reporting and inclusion thresholds. GA thresholds are generally considered to be the reliable curated thresholds defining family membership; for example, in Pfam, these thresholds define what gets included in Pfam Full alignments based on searches with Pfam Seed models.

--cut_nc Use the NC (noise cutoff) bit score thresholds in the model to set per-sequence (NC1) and per-domain (NC2) reporting and inclusion thresholds. NC thresholds are generally considered to be the score of the highest-scoring known false positive.

--cut_tc Use the NC (trusted cutoff) bit score thresholds in the model to set per-sequence (TC1) and per-domain (TC2) reporting and inclusion thresholds. TC thresholds are generally considered to be the score of the lowest-scoring known true positive that is above all known false positives.

Control of the Acceleration Pipeline

HMMER3 searches are accelerated in a three-step filter pipeline: the MSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm. There is also a bias filter step between MSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max Turn off all filters, including the bias filter, and run full Forward/Backward postprocessing on every target. This increases sensitivity somewhat, at a large cost in speed.

--F1 $<x>$ Set the P-value threshold for the MSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Other Options

--nonull2   Turn off the null2 score corrections for biased composition.

-Z $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-sequence E-value calculations, rather than the actual number of targets seen.

--domZ $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-domain conditional E-value calculations, rather than the number of targets that passed the reporting thresholds.

--seed $<n>$ Set the random number seed to $<n>$. Some steps in postprocessing require Monte Carlo simulation. The default is to use a fixed seed (42), so that results are exactly reproducible. Any other positive integer will give different (but also reproducible) results. A choice of 0 uses an arbitrarily chosen seed.

--qformat $<s>$ Assert that the query sequence file is in format $<s>$. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*.

--cpu $<n>$ Set the number of parallel worker threads to $<n>$. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting $<n>$ higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HM-MER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off for your site or machine for some reason.

--stall   For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--mpi   Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

`hmmsearch` - search profile(s) against a sequence database

Synopsis

hmmsearch *[options]* <*hmmfile*> <*seqdb*>

Description

hmmsearch is used to search one or more profiles against a sequence database. For each profile in <*hmmfile*>, use that query profile to search the target database of sequences in <*seqdb*>, and output ranked lists of the sequences with the most significant matches to the profile. To build profiles from multiple alignments, see **hmmbuild**.

Either the query <*hmmfile*> or the target <*seqdb*> may be '-' (a dash character), in which case the query profile or target database input will be read from a <stdin> pipe instead of from a file. Only one input source can come through <stdin>, not both. An exception is that if the <*hmmfile*> contains more than one profile query, then <*seqdb*> cannot come from <stdin>, because we can't rewind the streaming target database to search it with another profile.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** and **--domtblout** options save output in simple tabular formats that are concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

> **-h** Help; print a brief reminder of command line usage and all available options.

Options for Controlling Output

> **-o** <*f*> Direct the main human-readable output to a file <*f*> instead of the default stdout.

> **-A** <*f*> Save a multiple alignment of all significant hits (those satisfying *inclusion thresholds*) to the file <*f*>.

> **--tblout** <*f*> Save a simple tabular (space-delimited) file summarizing the per-target output, with one data line per homologous target sequence found.

> **--domtblout** <*f*> Save a simple tabular (space-delimited) file summarizing the per-domain output, with one data line per homologous domain detected in a query sequence for each homologous model.

> **--acc** Use accessions instead of names in the main output, where available for profiles and/or sequences.

> **--noali** Omit the alignment section from the main output. This can greatly reduce the output volume.

> **--notextw** Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

> **--textw** <*n*> Set the main output's line length limit to <*n*> characters per line. The default is 120.

Options Controlling Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--domtblout*). Sequence hits and domain hits are ranked by statistical significance (E-value) and output is generated in two sections called per-target and per-domain output. In per-target output, by default, all sequence hits with an E-value $<= 10$ are reported. In the per-domain output, for each target that has passed per-target reporting thresholds, all domains satisfying per-domain reporting thresholds are reported. By default, these are domains with conditional E-values of $<= 10$. The following options allow you to change the default E-value reporting thresholds, or to use bit score thresholds instead.

-E $<x>$ In the per-target output, report target sequences with an E-value of $<= <x>$. The default is 10.0, meaning that on average, about 10 false positives will be reported per query, so you can see the top of the noise and decide for yourself if it's really noise.

-T $<x>$ Instead of thresholding per-profile output on E-value, instead report target sequences with a bit score of $>= <x>$.

--domE $<x>$ In the per-domain output, for target sequences that have already satisfied the per-profile reporting threshold, report individual domains with a conditional E-value of $<= <x>$. The default is 10.0. A conditional E-value means the expected number of additional false positive domains in the smaller search space of those comparisons that already satisfied the per-target reporting threshold (and thus must have at least one homologous domain already).

--domT $<x>$ Instead of thresholding per-domain output on E-value, instead report domains with a bit score of $>= <x>$.

Options for Inclusion Thresholds

Inclusion thresholds are stricter than reporting thresholds. Inclusion thresholds control which hits are considered to be reliable enough to be included in an output alignment or a subsequent search round, or marked as significant ("!") as opposed to questionable ("?") in domain output.

--incE $<x>$ Use an E-value of $<= <x>$ as the per-target inclusion threshold. The default is 0.01, meaning that on average, about 1 false positive would be expected in every 100 searches with different query sequences.

--incT $<x>$ Instead of using E-values for setting the inclusion threshold, instead use a bit score of $>= <x>$ as the per-target inclusion threshold. By default this option is unset.

--incdomE $<x>$ Use a conditional E-value of $<= <x>$ as the per-domain inclusion threshold, in targets that have already satisfied the overall per-target inclusion threshold. The default is 0.01.

--incdomT $<x>$ Instead of using E-values, use a bit score of $>= <x>$ as the per-domain inclusion threshold.

Options for Model-specific Score Thresholding

Curated profile databases may define specific bit score thresholds for each profile, superseding any thresholding based on statistical significance alone. To use these options, the profile must contain the appropriate (GA, TC, and/or NC) optional score threshold annotation; this is picked up by **hmmbuild** from Stockholm format alignment files. Each thresholding option has two scores: the per-sequence threshold $<x1>$ and the per-domain threshold $<x2>$ These act as if **-T**$<x1>$ **--incT**$<x1>$ **--domT**$<x2>$ **--incdomT**$<x2>$ has been applied specifically using each model's curated thresholds.

--cut_ga Use the GA (gathering) bit scores in the model to set per-sequence (GA1) and per-domain (GA2) reporting and inclusion thresholds. GA thresholds are generally considered to be the reliable curated thresholds defining family membership; for example, in Pfam, these thresholds define what gets included in Pfam Full alignments based on searches with Pfam Seed models.

--cut_nc Use the NC (noise cutoff) bit score thresholds in the model to set per-sequence (NC1) and per-domain (NC2) reporting and inclusion thresholds. NC thresholds are generally considered to be the score of the highest-scoring known false positive.

--cut_tc Use the TC (trusted cutoff) bit score thresholds in the model to set per-sequence (TC1) and per-domain (TC2) reporting and inclusion thresholds. TC thresholds are generally considered to be the score of the lowest-scoring known true positive that is above all known false positives.

Options Controlling the Acceleration Pipeline

HMMER3 searches are accelerated in a three-step filter pipeline: the MSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm. There is also a bias filter step between MSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max Turn off all filters, including the bias filter, and run full Forward/Backward postprocessing on every target. This increases sensitivity somewhat, at a large cost in speed.

--F1 $<x>$ Set the P-value threshold for the MSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Other Options

    --nonull2   Turn off the null2 score corrections for biased composition.

    -Z $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-sequence E-value calculations, rather than the actual number of targets seen.

    --domZ $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-domain conditional E-value calculations, rather than the number of targets that passed the reporting thresholds.

    --seed $<n>$ Set the random number seed to $<n>$. Some steps in postprocessing require Monte Carlo simulation. The default is to use a fixed seed (42), so that results are exactly reproducible. Any other positive integer will give different (but also reproducible) results. A choice of 0 uses a randomly chosen seed.

    --tformat $<s>$ Assert that the target sequence database file is in format $<s>$. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*. The default is to autodetect the format of the file.

    --cpu $<n>$ Set the number of parallel worker threads to $<n>$. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting $<n>$ higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off at compile-time for your site or machine for some reason.

    --stall   For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

    --mpi   Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

`hmmsim` - collect score distributions on random sequences

Synopsis

hmmsim *[options]* *<hmmfile>*

Description

The **hmmsim** program generates random sequences, scores them with the model(s) in *<hmmfile>*, and outputs various sorts of histograms, plots, and fitted distributions for the resulting scores.

hmmsim is not a mainstream part of the HMMER package. Most users would have no reason to use it. It is used to develop and test the statistical methods used to determine P-values and E-values in HMMER3. For example, it was used to generate most of the results in a 2008 paper on H3's local alignment statistics (PLoS Comp Bio 4:e1000069, 2008; http://www.ploscompbiol.org/doi/pcbi.1000069).

Because it is a research testbed, you should not expect it to be as robust as other programs in the package. For example, options may interact in weird ways; we haven't tested nor tried to anticipate all different possible combinations.

The main task is to fit a maximum likelihood Gumbel distribution to Viterbi scores or an maximum likelihood exponential tail to high-scoring Forward scores, and to test that these fitted distributions obey the conjecture that lambda   log_2 for both the Viterbi Gumbel and the Forward exponential tail.

The output is a table of numbers, one row for each model. Four different parametric fits to the score data are tested: (1) maximum likelihood fits to both location (mu/tau) and slope (lambda) parameters; (2) assuming lambda=log_2, maximum likelihood fit to the location parameter only; (3) same but assuming an edge-corrected lambda, using current procedures in H3 [Eddy, 2008]; and (4) using both parameters determined by H3's current procedures. The standard simple, quick and dirty statistic for goodness-of-fit is 'E@10', the calculated E-value of the 10th ranked top hit, which we expect to be about 10.

In detail, the columns of the output are:

name	Name of the model.
tailp	Fraction of the highest scores used to fit the distribution. For Viterbi, MSV, and Hybrid scores, this defaults to 1.0 (a Gumbel distribution is fitted to all the data). For Forward scores, this defaults to 0.02 (an exponential tail is fitted to the highest 2% scores).
mu/tau	Location parameter for the maximum likelihood fit to the data.
lambda	Slope parameter for the maximum likelihood fit to the data.
E@10	The E-value calculated for the 10th ranked high score ('E@10') using the ML mu/tau and lambda. By definition, this expected to be about 10, if E-value estimation were accurate.
mufix	Location parameter, for a maximum likelihood fit with a known (fixed) slope parameter lambda of log_2 (0.693).
E@10fix	The E-value calculated for the 10th ranked score using mufix and the expected lambda = log_2 = 0.693.
mufix2	Location parameter, for a maximum likelihood fit with an edge-effect-corrected lambda.

E@10fix2 The E-value calculated for the 10th ranked score using mufix2 and the edge-effect-corrected lambda.

pmu Location parameter as determined by H3's estimation procedures.

plambda Slope parameter as determined by H3's estimation procedures.

pE@10 The E-value calculated for the 10th ranked score using pmu, plambda.

At the end of this table, one more line is printed, starting with # and summarizing the overall CPU time used by the simulations.

Some of the optional output files are in xmgrace xy format. xmgrace is powerful and freely available graph-plotting software.

Miscellaneous Options

-h Help; print a brief reminder of command line usage and all available options.

-a Collect expected Viterbi alignment length statistics from each simulated sequence. This only works with Viterbi scores (the default; see *--vit*). Two additional fields are printed in the output table for each model: the mean length of Viterbi alignments, and the standard deviation.

-v (Verbose). Print the scores too, one score per line.

-L $<n>$ Set the length of the randomly sampled (nonhomologous) sequences to $<n>$. The default is 100.

-N $<n>$ Set the number of randomly sampled sequences to $<n>$. The default is 1000.

--mpi Run in MPI parallel mode, under **mpirun**. It is parallelized at the level of sending one profile at a time to an MPI worker process, so parallelization only helps if you have more than one profile in the $<hmmfile>$, and you want to have at least as many profiles as MPI worker processes. (Only available if optional MPI support was enabled at compile-time.)

Options Controlling Output

-o $<f>$ Save the main output table to a file $<f>$ rather than sending it to stdout.

--afile $<f>$ When collecting Viterbi alignment statistics (the *-a* option), for each sampled sequence, output two fields per line to a file $<f>$: the length of the optimal alignment, and the Viterbi bit score. Requires that the *-a* option is also used.

--efile $<f>$ Output a rank vs. E-value plot in XMGRACE xy format to file $<f>$. The x-axis is the rank of this sequence, from highest score to lowest; the y-axis is the E-value calculated for this sequence. E-values are calculated using H3's default procedures (i.e. the pmu, plambda parameters in the output table). You expect a rough match between rank and E-value if E-values are accurately estimated.

--ffile $<f>$ Output a "filter power" file to $<f>$: for each model, a line with three fields: model name, number of sequences passing the P-value threshold, and fraction of sequences passing the P-value threshold. See *--pthresh* for setting the P-value threshold, which defaults to 0.02 (the default MSV filter threshold in H3). The P-values are as determined by H3's default procedures (the pmu,plambda parameters in the output table). If all is well, you expect to see filter power equal to the predicted P-value setting of the threshold.

--pfile $<f>$ Output cumulative survival plots (P(S>x)) to file $<f>$ in XMGRACE xy format. There are three plots: (1) the observed score distribution; (2) the maximum likelihood fitted distribution; (3) a maximum likelihood fit to the location parameter (mu/tau) while assuming lambda=log_2.

--xfile $<f>$ Output the bit scores as a binary array of double-precision floats (8 bytes per score) to file $<f>$. Programs like Easel's **esl-histplot** can read such binary files. This is useful when generating extremely large sample sizes.

Options Controlling Model Configuration (mode)

H3 only uses multihit local alignment (*--fs* mode), and this is where we believe the statistical fits. Unihit local alignment scores (Smith/Waterman; *--sw* mode) also obey our statistical conjectures. Glocal alignment statistics (either multihit or unihit) are still not adequately understood nor adequately fitted.

--fs   Collect multihit local alignment scores. This is the default. alignment as 'fragment search mode'.

--sw   Collect unihit local alignment scores. The H3 J state is disabled. alignment as 'Smith/Waterman search mode'.

--ls   Collect multihit glocal alignment scores. In glocal (global/local) alignment, the entire model must align, to a subsequence of the target. The H3 local entry/exit transition probabilities are disabled. 'ls' comes from HMMER2's historical terminology for multihit local alignment as 'local search mode'.

--s   Collect unihit glocal alignment scores. Both the H3 J state and local entry/exit transition probabilities are disabled. 's' comes from HMMER2's historical terminology for unihit glocal alignment.

Options Controlling Scoring Algorithm

--vit   Collect Viterbi maximum likelihood alignment scores. This is the default.

--fwd   Collect Forward log-odds likelihood scores, summed over alignment ensemble.

--hyb   Collect 'Hybrid' scores, as described in papers by Yu and Hwa (for instance, Bioinformatics 18:864, 2002). These involve calculating a Forward matrix and taking the maximum cell value. The number itself is statistically somewhat unmotivated, but the distribution is expected be a well-behaved extreme value distribution (Gumbel).

--msv   Collect MSV (multiple ungapped segment Viterbi) scores, using H3's main acceleration heuristic.

--fast   For any of the above options, use H3's optimized production implementation (using SIMD vectorization). The default is to use the implementations sacrifice a small amount of numerical precision. This can introduce confounding noise into statistical simulations and fits, so when one gets super-concerned about exact details, it's better to be able to factor that source of noise out.

Options Controlling Fitted Tail Masses for Forward

In some experiments, it was useful to fit Forward scores to a range of different tail masses, rather than just one. These options provide a mechanism for fitting an evenly-spaced range of different tail masses. For each different tail mass, a line is generated in the output.

--tmin $<x>$ Set the lower bound on the tail mass distribution. (The default is 0.02 for the default single tail mass.)

--tmax $<x>$ Set the upper bound on the tail mass distribution. (The default is 0.02 for the default single tail mass.)

--tpoints $<n>$ Set the number of tail masses to sample, starting from *--tmin* and ending at *--tmax*. (The default is 1, for the default 0.02 single tail mass.)

--tlinear   Sample a range of tail masses with uniform linear spacing. The default is to use uniform logarithmic spacing.

Options Controlling H3 Parameter Estimation Methods

H3 uses three short random sequence simulations to estimating the location parameters for the expected score distributions for MSV scores, Viterbi scores, and Forward scores. These options allow these simulations to be modified.

--EmL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for MSV E-values. Default is 200.

--EmN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for MSV E-values. Default is 200.

--EvL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for Viterbi E-values. Default is 200.

--EvN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for Viterbi E-values. Default is 200.

--EfL $<n>$ Sets the sequence length in simulation that estimates the location parameter tau for Forward E-values. Default is 100.

--EfN $<n>$ Sets the number of sequences in simulation that estimates the location parameter tau for Forward E-values. Default is 200.

--Eft $<x>$ Sets the tail mass fraction to fit in the simulation that estimates the location parameter tau for Forward evalues. Default is 0.04.

Debugging Options

--stall For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--seed *<n>* Set the random number seed to *<n>*. The default is 0, which makes the random number generator use an arbitrary seed, so that different runs of **hmmsim** will almost certainly generate a different statistical sample. For debugging, it is useful to force reproducible results, by fixing a random number seed.

Experimental Options

These options were used in a small variety of different exploratory experiments.

--bgflat Set the background residue distribution to a uniform distribution, both for purposes of the null model used in calculating scores, and for generating the random sequences. The default is to use a standard amino acid background frequency distribution.

--bgcomp Set the background residue distribution to the mean composition of the profile. This was used in exploring some of the effects of biased composition.

--x-no-lengthmodel Turn the H3 target sequence length model off. Set the self-transitions for N,C,J and the null model to 350/351 instead; this emulates HMMER2. Not a good idea in general. This was used to demonstrate one of the main H2 vs. H3 differences.

--nu *<x>* Set the nu parameter for the MSV algorithm -- the expected number of ungapped local alignments per target sequence. The default is 2.0, corresponding to a E->J transition probability of 0.5. This was used to test whether varying nu has significant effect on result (it doesn't seem to, within reason). This option only works if *--msv* is selected (it only affects MSV), and it will not work with *--fast* (because the optimized implementations are hardwired to assume nu=2.0).

--pthresh *<x>* Set the filter P-value threshold to use in generating filter power files with *--ffile*. The default is 0.02 (which would be appropriate for testing MSV scores, since this is the default MSV filter threshold in H3's acceleration pipeline.) Other appropriate choices (matching defaults in the acceleration pipeline) would be 0.001 for Viterbi, and 1e-5 for Forward.

`hmmstat` - display summary statistics for a profile file

Synopsis

hmmstat *[options]* *<hmmfile>*

Description

The **hmmstat** utility prints out a tabular file of summary statistics for each profile in *<hmmfile>*.

<hmmfile> may be '-' (a dash character), in which case profiles are read from a *<stdin>* pipe instead of from a file.

The columns are:

idx	The index of this profile, numbering each on in the file starting from 1.
name	The name of the profile.
accession	The optional accession of the profile, or "-" if there is none.
nseq	The number of sequences that the profile was estimated from.
eff_nseq	The effective number of sequences that the profile was estimated from, after HMMER applied an effective sequence number calculation such as the default entropy weighting.
M	The length of the model in consensus residues (match states).
relent	Mean relative entropy per match state, in bits. This is the expected (mean) score per consensus position. This is what the default entropy-weighting method for effective sequence number estimation focuses on, so for default HMMER3 models, you expect this value to reflect the default target for entropy-weighting.
info	Mean information content per match state, in bits. Probably not useful. Information content is a slightly different calculation than relative entropy.
p relE	Mean positional relative entropy, in bits. This is a fancier version of the per-match-state relative entropy, taking into account the transition (insertion/deletion) probabilities; it may be a more accurate estimation of the average score contributed per model consensus position.
compKL	Kullback-Leibler distance between the model's overall average residue composition and the default background frequency distribution. The higher this number, the more biased the residue composition of the profile is. Highly biased profiles can slow the HMMER3 acceleration pipeline, by causing too many nonhomologous sequences to pass the filters.

Options

-h	Help; print a brief reminder of command line usage and all available options.

`jackhmmer` - iteratively search sequence(s) against a protein database

Synopsis

jackhmmer *[options]* <*seqfile*> <*seqdb*>

Description

jackhmmer iteratively searches each query sequence in <*seqfile*> against the target sequence(s) in <*seqdb*>. The first iteration is identical to a **phmmer** search. For the next iteration, a multiple alignment of the query together with all target sequences satisfying *inclusion thresholds* is assembled, a profile is constructed from this alignment (identical to using **hmmbuild** on the alignment), and profile search of the <*seqdb*> is done (identical to an **hmmsearch** with the profile).

The query <*seqfile*> may be '-' (a dash character), in which case the query sequences are read from a <stdin> pipe instead of from a file. The <*seqdb*> cannot be read from a <stdin> stream, because **jackhmmer** needs to do multiple passes over the database.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** and **--domtblout** options save output in simple tabular formats that are concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

-h	Help; print a brief reminder of command line usage and all available options.
-N <*n*>	Set the maximum number of iterations to <*n*>. The default is 5. If N=1, the result is equivalent to a **phmmer** search.

Options Controlling Output

By default, output for each iteration appears on stdout in a somewhat human readable, somewhat parseable format. These options allow redirecting that output or saving additional kinds of output to files, including checkpoint files for each iteration.

-o <*f*>	Direct the human-readable output to a file <*f*>.
-A <*f*>	After the final iteration, save an annotated multiple alignment of all hits satisfying inclusion thresholds (also including the original query) to <*f*> in Stockholm format.
--tblout <*f*>	After the final iteration, save a tabular summary of top sequence hits to <*f*> in a readily parseable, columnar, whitespace-delimited format.
--domtblout <*f*>	After the final iteration, save a tabular summary of top domain hits to <*f*> in a readily parseable, columnar, whitespace-delimited format.
--chkhmm <*prefix*>	At the start of each iteration, checkpoint the query HMM, saving it to a file named <*prefix*>-<*n*>.hmm where <*n*> is the iteration number (from 1..N).

--chkali *<prefix>*   At the end of each iteration, checkpoint an alignment of all domains satisfying inclusion thresholds (e.g. what will become the query HMM for the next iteration), saving it to a file named *<checkpoint file prefix>-<n>.sto* in Stockholm format, where *<n>* is the iteration number (from 1..N).

--acc   Use accessions instead of names in the main output, where available for profiles and/or sequences.

--noali   Omit the alignment section from the main output. This can greatly reduce the output volume.

--notextw   Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

--textw *<n>*   Set the main output's line length limit to *<n>* characters per line. The default is 120.

Options Controlling Single Sequence Scoring (first Iteration)

By default, the first iteration uses a search model constructed from a single query sequence. This model is constructed using a standard 20x20 substitution matrix for residue probabilities, and two additional parameters for position-independent gap open and gap extend probabilities. These options allow the default single-sequence scoring parameters to be changed.

--popen *<x>*   Set the gap open probability for a single sequence query model to *<x>*. The default is 0.02. *<x>* must be $>= 0$ and < 0.5.

--pextend *<x>*   Set the gap extend probability for a single sequence query model to *<x>*. The default is 0.4. *<x>* must be $>= 0$ and < 1.0.

--mx *<s>*   Obtain residue alignment probabilities from the built-in substitution matrix named *<s>*. Several standard matrices are built-in, and do not need to be read from files. The matrix name *<s>* can be PAM30, PAM70, PAM120, PAM240, BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, or BLOSUM90. Only one of the **--mx** and **--mxfile** options may be used.

--mxfile *<mxfile>*   Obtain residue alignment probabilities from the substitution matrix in file *<mxfile>*. The default score matrix is BLOSUM62 (this matrix is internal to HMMER and does not have to be available as a file). The format of a substitution matrix *<mxfile>* is the standard format accepted by BLAST, FASTA, and other sequence analysis software.

Options Controlling Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--domtblout*). In each iteration, sequence hits and domain hits are ranked by statistical significance (E-value) and output is generated in two sections called per-target and per-domain output. In per-target output, by default, all sequence hits with an E-value <= 10 are reported. In the per-domain output, for each target that has passed per-target reporting thresholds, all domains satisfying per-domain reporting thresholds are reported. By default, these are domains with conditional E-values of <= 10. The following options allow you to change the default E-value reporting thresholds, or to use bit score thresholds instead.

-E $<x>$ Report sequences with E-values $<= <x>$ in per-sequence output. The default is 10.0.

-T $<x>$ Use a bit score threshold for per-sequence output instead of an E-value threshold (any setting of **-E** is ignored). Report sequences with a bit score of $>= <x>$. By default this option is unset.

-Z $<x>$ Declare the total size of the database to be $<x>$ sequences, for purposes of E-value calculation. Normally E-values are calculated relative to the size of the database you actually searched (e.g. the number of sequences in *target_seqdb*). In some cases (for instance, if you've split your target sequence database into multiple files for parallelization of your search), you may know better what the actual size of your search space is.

--domE $<x>$ Report domains with conditional E-values $<= <x>$ in per-domain output, in addition to the top-scoring domain per significant sequence hit. The default is 10.0.

--domT $<x>$ Use a bit score threshold for per-domain output instead of an E-value threshold (any setting of **--domT** is ignored). Report domains with a bit score of $>= <x>$ in per-domain output, in addition to the top-scoring domain per significant sequence hit. By default this option is unset.

--domZ $<x>$ Declare the number of significant sequences to be $<x>$ sequences, for purposes of conditional E-value calculation for additional domain significance. Normally conditional E-values are calculated relative to the number of sequences passing per-sequence reporting threshold.

Options Controlling Inclusion Thresholds

Inclusion thresholds control which hits are included in the multiple alignment and profile constructed for the next search iteration. By default, a sequence must have a per-sequence E-value of $<= 0.001$ (see **-E** option) to be included, and any additional domains in it besides the top-scoring one must have a conditional E-value of $<= 0.001$ (see **--domE** option). The difference between reporting thresholds and inclusion thresholds is that inclusion thresholds control which hits actually get used in the next iteration (or the final output multiple alignment if the *-A* option is used), whereas reporting thresholds control what you see in output. Reporting thresholds are generally more loose so you can see borderline hits in the top of the noise that might be of interest.

--incE $<x>$ Include sequences with E-values $<= <x>$ in subsequent iteration or final alignment output by *-A*. The default is 0.001.

--incT $<x>$ Use a bit score threshold for per-sequence inclusion instead of an E-value threshold (any setting of **--incE** is ignored). Include sequences with a bit score of $>= <x>$. By default this option is unset.

--incdomE $<x>$ Include domains with conditional E-values $<= <x>$ in subsequent iteration or final alignment output by *-A*, in addition to the top-scoring domain per significant sequence hit. The default is 0.001.

--incdomT $<x>$ Use a bit score threshold for per-domain inclusion instead of an E-value threshold (any setting of **--incT** is ignored). Include domains with a bit score of $>= <x>$. By default this option is unset.

Options Controlling Acceleration Heuristics

HMMER3 searches are accelerated in a three-step filter pipeline: the MSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm, slowest but most accurate. There is also a bias filter step between MSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Essentially the only free parameters that control HMMER's heuristic filters are the P-value thresholds controlling the expected fraction of nonhomologous sequences that pass the filters. Setting the default thresholds higher will pass a higher proportion of nonhomologous sequence, increasing sensitivity at the expense of speed; conversely, setting lower P-value thresholds will pass a smaller proportion, decreasing sensitivity and increasing speed. Setting a filter's P-value threshold to 1.0 means it will passing all sequences, and effectively disables the filter. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max   Maximum sensitivity. Turn off all filters, including the bias filter, and run full Forward/Backward postprocessing on every target. This increases sensitivity slightly, at a large cost in speed.

--F1 $<x>$ First filter threshold; set the P-value threshold for the MSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Second filter threshold; set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Third filter threshold; set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias   Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Options Controlling Profile Construction (later Iterations)

These options control how consensus columns are defined in multiple alignments when building profiles. By default, **jackhmmer** always includes your original query sequence in the alignment result at every iteration, and consensus positions are defined by that query sequence: that is, a default **jackhmmer** profile is always the same length as your original query, at every iteration.

--fast   Define consensus columns as those that have a fraction $>=$ **symfrac** of residues as opposed to gaps. (See below for the **--symfrac** option.) Although this is the default profile construction option elsewhere (in **hmmbuild**, in particular), it may have undesirable effects in **jackhmmer**, because a profile could iteratively walk in sequence space away from your original query, leaving few or no consensus columns corresponding to its residues.

--hand Define consensus columns in next profile using reference annotation to the multiple alignment. **jackhmmer** propagates reference annotation from the previous profile to the multiple alignment, and thence to the next profile. This is the default.

--symfrac *<x>* Define the residue fraction threshold necessary to define a consensus column when using the **--fast** option. The default is 0.5. The symbol fraction in each column is calculated after taking relative sequence weighting into account, and ignoring gap characters corresponding to ends of sequence fragments (as opposed to internal insertions/deletions). Setting this to 0.0 means that every alignment column will be assigned as consensus, which may be useful in some cases. Setting it to 1.0 means that only columns that include 0 gaps (internal insertions/deletions) will be assigned as consensus.

--fragthresh *<x>* We only want to count terminal gaps as deletions if the aligned sequence is known to be full-length, not if it is a fragment (for instance, because only part of it was sequenced). HMMER uses a simple rule to infer fragments: if the sequence length L is less than or equal to a fraction *<x>* times the alignment length in columns, then the sequence is handled as a fragment. The default is 0.5. Setting **--fragthresh***0* will define no (nonempty) sequence as a fragment; you might want to do this if you know you've got a carefully curated alignment of full-length sequences. Setting **--fragthresh***1* will define all sequences as fragments; you might want to do this if you know your alignment is entirely composed of fragments, such as translated short reads in metagenomic shotgun data.

Options Controlling Relative Weights

Whenever a profile is built from a multiple alignment, HMMER uses an ad hoc sequence weighting algorithm to downweight closely related sequences and upweight distantly related ones. This has the effect of making models less biased by uneven phylogenetic representation. For example, two identical sequences would typically each receive half the weight that one sequence would (and this is why **jackhmmer** isn't concerned about always including your original query sequence in each iteration's alignment, even if it finds it again in the database you're searching). These options control which algorithm gets used.

--wpb Use the Henikoff position-based sequence weighting scheme [Henikoff and Henikoff, J. Mol. Biol. 243:574, 1994]. This is the default.

--wgsc Use the Gerstein/Sonnhammer/Chothia weighting algorithm [Gerstein et al, J. Mol. Biol. 235:1067, 1994].

--wblosum Use the same clustering scheme that was used to weight data in calculating BLOSUM subsitution matrices [Henikoff and Henikoff, Proc. Natl. Acad. Sci 89:10915, 1992]. Sequences are single-linkage clustered at an identity threshold (default 0.62; see *--wid*) and within each cluster of c sequences, each sequence gets relative weight 1/c.

--wnone No relative weights. All sequences are assigned uniform weight.

--wid *<x>* Sets the identity threshold used by single-linkage clustering when using *--wblosum*. Invalid with any other weighting scheme. Default is 0.62.

Options Controlling Effective Sequence Number

After relative weights are determined, they are normalized to sum to a total effective sequence number, *eff_nseq*. This number may be the actual number of sequences in the alignment, but it is almost always smaller than that. The default entropy weighting method *(--eent)* reduces the effective sequence number to reduce the information content (relative entropy, or average expected score on true homologs) per consensus position. The target relative entropy is controlled by a two-parameter function, where the two parameters are settable with *--ere* and *--esigma*.

--eent Adjust effective sequence number to achieve a specific relative entropy per position (see *--ere*). This is the default.

--eclust Set effective sequence number to the number of single-linkage clusters at a specific identity threshold (see *--eid*). This option is not recommended; it's for experiments evaluating how much better **--eent** is.

--enone Turn off effective sequence number determination and just use the actual number of sequences. One reason you might want to do this is to try to maximize the relative entropy/position of your model, which may be useful for short models.

--eset $<x>$ Explicitly set the effective sequence number for all models to $<x>$.

--ere $<x>$ Set the minimum relative entropy/position target to $<x>$. Requires **--eent**. Default depends on the sequence alphabet; for protein sequences, it is 0.59 bits/position.

--esigma $<x>$ Sets the minimum relative entropy contributed by an entire model alignment, over its whole length. This has the effect of making short models have higher relative entropy per position than *--ere* alone would give. The default is 45.0 bits.

--eid $<x>$ Sets the fractional pairwise identity cutoff used by single linkage clustering with the **--eclust** option. The default is 0.62.

Options Controlling Priors

In profile construction, by default, weighted counts are converted to mean posterior probability parameter estimates using mixture Dirichlet priors. Default mixture Dirichlet prior parameters for protein models and for nucleic acid (RNA and DNA) models are built in. The following options allow you to override the default priors. **--pnone** Don't use any priors. Probability parameters will simply be the observed frequencies, after relative sequence weighting. **--plaplace** Use a Laplace +1 prior in place of the default mixture Dirichlet prior.

Options Controlling E-value Calibration

Estimating the location parameters for the expected score distributions for MSV filter scores, Viterbi filter scores, and Forward scores requires three short random sequence simulations.

--EmL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EmN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EvL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EvN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EfL $<n>$ Sets the sequence length in simulation that estimates the location parameter tau for Forward E-values. Default is 100.

--EfN $<n>$ Sets the number of sequences in simulation that estimates the location parameter tau for Forward E-values. Default is 200.

--Eft $<x>$ Sets the tail mass fraction to fit in the simulation that estimates the location parameter tau for Forward evalues. Default is 0.04.

Other Options

--nonull2   Turn off the null2 score corrections for biased composition.

-Z $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-sequence E-value calculations, rather than the actual number of targets seen.

--domZ $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-domain conditional E-value calculations, rather than the number of targets that passed the reporting thresholds.

--seed $<n>$ Seed the random number generator with $<n>$, an integer $>= 0$. If $<n>$ is >0, any stochastic simulations will be reproducible; the same command will give the same results. If $<n>$ is 0, the random number generator is seeded arbitrarily, and stochastic simulations will vary from run to run of the same command. The default seed is 42.

--qformat $<s>$ Declare that the input *query_seqfile* is in format $<s>$. Accepted sequence file formats include FASTA, EMBL, GenBank, DDBJ, UniProt, Stockholm, and SELEX. Default is to autodetect the format of the file.

--tformat $<s>$ Declare that the input *target_seqdb* is in format $<s>$. Accepted sequence file formats include FASTA, EMBL, GenBank, DDBJ, UniProt, Stockholm, and SELEX. Default is to autodetect the format of the file.

--cpu $<n>$ Set the number of parallel worker threads to $<n>$. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting $<n>$ higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off at compile-time for your site or machine for some reason.

--stall   For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--mpi   Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

`makehmmerdb` - build a HMMER binary database file from a sequence file

Synopsis

makehmmerdb *[options]* *⟨seqfile⟩* *⟨binaryfile⟩*

Description

makehmmerdb is used to create a binary file from a DNA sequence file. This binary file may be used as a target database for the DNA search tool **nhmmer**.**Using**default**settings**in *nhmmer*, this yields a roughly 10-fold acceleration with small loss of sensitivity on benchmarks. (This method has been extensively tested, but should still be treated as somewhat experimental.)

Options

    -h Help; print a brief reminder of command line usage and all available options.

Other Options

    --informat *⟨s⟩* Assert that the sequence database file is in format *⟨s⟩*. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*. The default is to autodetect the format of the file.

    --bin_length *⟨n⟩* Bin length. The binary file depends on a data structure called the FM index, which organizes a permuted copy of the sequence in bins of length *⟨n⟩*. Longer bin length will lead to smaller files (because data is captured about each bin) and possibly slower query time. The default is 256. Much more than 512 may lead to notable reduction in speed.

    --sa_freq *⟨n⟩* Suffix array sample rate. The FM index structure also samples from the underlying suffix array for the sequence database. More frequent sampling (smaller value for *⟨n⟩*) will yield larger file size and faster search (until file size becomes large enough to cause I/O to be a bottleneck). The default value is 8. Must be a power of 2.

    --block_size *⟨n⟩* The input sequence is broken into blocks of size *⟨n⟩* million letters. An FM index is built for each block, rather than building an FM index for the entire sequence database. Default is 50. Larger blocks do not seem to yield substantial speed increase.

`nhmmer` - search DNA/RNA queries against a DNA/RNA sequence database

Synopsis

nhmmer [options] <*queryfile*> <*seqdb*>

Description

nhmmer is used to search one or more nucleotide queries against a nucleotide sequence database. For each query in <*queryfile*>, use that query to search the target database of sequences in <*seqdb*>, and output a ranked list of the hits with the most significant matches to the query. A query may be either a profile model built using **hmmbuild**, a sequence alignment, or a single sequence. Sequence based queries can be in a number of formats (see **--qformat**), and can typically be autodetected. Note that only *Stockholm* format supports queries made up of more than one sequence alignment.

Either the query <*queryfile*> or the target <*seqdb*> may be '-' (a dash character), in which case the query file or target database input will be read from a <stdin> pipe instead of from a file. Only one input source can come through <stdin>, not both. If the query is sequence-based and passed via <stdin>, the **--qformat** flag must be used. If the <*queryfile*> contains more than one query, then <*seqdb*> cannot come from <stdin>, because we can't rewind the streaming target database to search it with another profile.

If the query is sequence-based, and not from <stdin>, a new file containing the HMM(s) built from the input(s) in <*queryfile*> may optionally be produced, with the filename set using the **--hmmout** flag.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** option saves output in a simple tabular format that is concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

      -h Help; print a brief reminder of command line usage and all available options.

Options for Controlling Output

      -o <*f*> Direct the main human-readable output to a file <*f*> instead of the default stdout.

      -A <*f*> Save a multiple alignment of all significant hits (those satisfying *inclusion thresholds*) to the file <*f*>.

      --tblout <*f*> Save a simple tabular (space-delimited) file summarizing the per-target output, with one data line per homologous target sequence found.

      --dfamtblout <*f*> Save a tabular (space-delimited) file summarizing the per-hit output, similar to **--tblout** but more succinct.

      --aliscoresout <*f*> Save to file a list of per-position scores for each hit. This is useful, for example, in identifying regions of high score density for use in resolving overlapping hits from different models.

      --hmmout <*f*> If <queryfile> is sequence-based, write the internally-computed HMM(s) to <*f*>.

--acc   Use accessions instead of names in the main output, where available for profiles and/or sequences.

--noali   Omit the alignment section from the main output. This can greatly reduce the output volume.

--notextw Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

--textw $<n>$ Set the main output's line length limit to $<n>$ characters per line. The default is 120.

Options Controlling Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--dfamtblout*). Hits are ranked by statistical significance (E-value).

-E $<x>$ Report target sequences with an E-value of $<= <x>$. The default is 10.0, meaning that on average, about 10 false positives will be reported per query, so you can see the top of the noise and decide for yourself if it's really noise.

-T $<x>$ Instead of thresholding output on E-value, instead report target sequences with a bit score of $>= <x>$.

Options for Inclusion Thresholds

Inclusion thresholds are stricter than reporting thresholds. Inclusion thresholds control which hits are considered to be reliable enough to be included in an output alignment or a subsequent search round, or marked as significant ("!") as opposed to questionable ("?") in hit output.

--incE $<x>$ Use an E-value of $<= <x>$ as the inclusion threshold. The default is 0.01, meaning that on average, about 1 false positive would be expected in every 100 searches with different query sequences.

--incT $<x>$ Instead of using E-values for setting the inclusion threshold, use a bit score of $>= <x>$ as the inclusion threshold. By default this option is unset.

Options for Model-specific Score Thresholding

Curated profile databases may define specific bit score thresholds for each profile, superseding any thresholding based on statistical significance alone. To use these options, the profile must contain the appropriate (GA, TC, and/or NC) optional score threshold annotation; this is picked up by **hmmbuild** from Stockholm format alignment files. For a nucleotide model, each thresholding option has a single per-hit threshold $<x>$ This acts as if **-T**$<x>$ **--incT**$<x>$ has been applied specifically using each model's curated thresholds.

--cut_ga Use the GA (gathering) bit score threshold in the model to set per-hit reporting and inclusion thresholds. GA thresholds are generally considered to be the reliable curated thresholds defining family membership; for example, in Dfam, these thresholds are applied when annotating a genome with a model of a family known to be found in that organism. They may allow for minimal expected false discovery rate.

--cut_nc Use the NC (noise cutoff) bit score threshold in the model to set per-hit reporting and inclusion thresholds. NC thresholds are less stringent than GA; in the context of Pfam, they are generally used to store the score of the highest-scoring known false positive.

--cut_tc Use the NC (trusted cutoff) bit score threshold in the model to set per-hit reporting and inclusion thresholds. TC thresholds are more stringent than GA, and are generally considered to be the score of the lowest-scoring known true positive that is above all known false positives; for example, in Dfam, these thresholds are applied when annotating a genome with a model of a family not known to be found in that organism.

Options Controlling the Acceleration Pipeline

HMMER3 searches are accelerated in a three-step filter pipeline: the scanning-SSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm. There is also a bias filter step between SSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max Turn off (nearly) all filters, including the bias filter, and run full Forward/Backward postprocessing on most of the target sequence. In contrast to **phmmer** and **hmm-search**, where this flag really does turn off the filters entirely, the *--max* flag in **nhmmer** sets the scanning-SSV filter threshold to 0.4, not 1.0. Use of this flag increases sensitivity somewhat, at a large cost in speed.

--F1 $<x>$ Set the P-value threshold for the SSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Options for Specifying the Alphabet

The alphabet type of the target database (DNA or RNA) is autodetected by default, by looking at the composition of the $<seqdb>$. Autodetection is normally quite reliable, but occasionally alphabet type may be ambiguous and autodetection can fail (for instance, when the first sequence starts with a run of ambiguous characters). To avoid this, or to increase robustness in automated analysis pipelines, you may specify the alphabet type of $<seqdb>$ with these options.

--dna   Specify that all sequences in <*seqdb*> are DNAs.

--rna   Specify that all sequences in <*seqdb*> are RNAs.

Options Controlling Seed Search Heuristic

When searching with *nhmmer*, one may optionally precompute a binary version of the target database, using *makehmmerdb*, then search against that database. Using default settings, this yields a roughly 10-fold acceleration with small loss of sensitivity on benchmarks. This is achieved using a heuristic method that searches for seeds (ungapped alignments) around which full processing is done. This is essentially a replacement to the SSV stage. (This method has been extensively tested, but should still be treated as somewhat experimental.) The following options only impact *nhmmer* if the value of **--tformat** is *hmmerdb*. Changing parameters for this seed-finding step will impact both speed and sensitivity - typically faster search leads to lower sensitivity.

--seed_max_depth <*n*>   The seed step requires that a seed reach a specified bit score in length no longer than <**n**>. By default, this value is 15. Longer seeds allow a greater chance of meeting the bit score threshold, leading to diminished filtering (greater sensitivity, slower run time).

--seed_sc_thresh <*x*>   The seed must reach score <**x**> (in bits). The default is 15.0 bits. A higher threshold increases filtering stringency, leading to faster run times and lower sensitivity.

--seed_sc_density <*x*>   Either all prefixes or all suffixes of a seed must have bit density (bits per aligned position) of at least <**x**>. The default is 0.8 bits/position. An increase in the density requirement leads to increased filtering stringency, thus faster run times and lower sensitivity.

--seed_drop_max_len <*n*>   A seed may not have a run of length <**n**> in which the score drops by *--seed_drop_lim* or more. Basically, this prunes seeds that go through long slightly-negative seed extensions. The default is 4. Increasing the limit causes (slightly) diminished filtering efficiency, thus slower run times and higher sensitivity. (minor tuning option)

--seed_drop_lim <*x*>   In a seed, there may be no run of length *--seed_drop_max_len* in which the score drops by *--seed_drop_lim*. The default is 0.3 bits. Larger numbers mean less filtering. (minor tuning option)

--seed_req_pos <*n*>   A seed must contain a run of at least <**n**> positive-scoring matches. The default is 5. Larger values mean increased filtering. (minor tuning option)

--seed_ssv_length <*n*>   After finding a short seed, an ungapped alignment is extended in both directions in an attempt to meet the *--F1* score threshold. The window through which this ungapped alignment extends is length <**n**>. The default is 70. Decreasing this value slightly reduces run time, at a small risk of reduced sensitivity. (minor tuning option)

Other Options

--tformat <*s*>   Assert that the target sequence database file is in format <*s*>. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, *afa*, and *hmmerfm*. The default is to autodetect the format of the file. The format *hmmerfm* indicates that the database file is a binary file produced using *makehmmerdb* (this format is not currently autodetected).

--qformat $<s>$ Declare that the input $<queryfile>$ is in format $<s>$. This is used when the query is sequence-based, rather than made up of profile model(s). Currently the accepted multiple alignment sequence file formats include Stockholm, Aligned FASTA, Clustal, NCBI PSI-BLAST, PHYLIP, Selex, and UCSC SAM A2M. Default is to autodetect the format of the file.

--nonull2   Turn off the null2 score corrections for biased composition.

-Z $<x>$ For the purposes of per-hit E-value calculations, Assert that the total size of the target database is $<x>$ million nucleotides, rather than the actual number of targets seen.

--seed $<n>$ Set the random number seed to $<n>$. Some steps in postprocessing require Monte Carlo simulation. The default is to use a fixed seed (42), so that results are exactly reproducible. Any other positive integer will give different (but also reproducible) results. A choice of 0 uses a randomly chosen seed.

--w_beta $<x>$ Window length tail mass. The upper bound, W, on the length at which nhmmer expects to find an instance of the model is set such that the fraction of all sequences generated by the model with length $>= W$ is less than $<x>$. The default is 1e-7. This flag may be used to override the value of W established for the model by **hmmbuild**, or when the query is sequence-based.

--w_length $<n>$ Override the model instance length upper bound, W, which is otherwise controlled by **--w_beta**. It should be larger than the model length. The value of W is used deep in the acceleration pipeline, and modest changes are not expected to impact results (though larger values of W do lead to longer run time). This flag may be used to override the value of W established for the model by **hmmbuild**, or when the query is sequence-based.

--toponly   Only search the top strand. By default both the query sequence and its reverse-complement are searched.

--bottomonly   Only search the bottom (reverse-complement) strand. By default both the query sequence and its reverse-complement are searched.

--cpu $<n>$ Set the number of parallel worker threads to $<n>$. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting $<n>$ higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off at compile-time for your site or machine for some reason.

--stall   For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--mpi   Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

`nhmmscan` - search nucleotide sequence(s) against a nucleotide profile

Synopsis

hmmscan *[options]* *<hmmdb>* *<seqfile>*

Description

nhmmscan is used to search nucleotide sequences against collections of nucleotide profiles. For each sequence in *<seqfile>*, use that query sequence to search the target database of profiles in *<hmmdb>*, and output ranked lists of the profiles with the most significant matches to the sequence.

The *<seqfile>* may contain more than one query sequence. It can be in FASTA format, or several other common sequence file formats (genbank, embl, and uniprot, among others), or in alignment file formats (stockholm, aligned fasta, and others). See the *--qformat* option for a complete list.

The *<hmmdb>* needs to be press'ed using **hmmpress** before it can be searched with **hmmscan**. This creates four binary files, suffixed .h3{fimp}.

The query *<seqfile>* may be '-' (a dash character), in which case the query sequences are read from a *<stdin>* pipe instead of from a file. The *<hmmdb>* cannot be read from a *<stdin>* stream, because it needs to have those four auxiliary binary files generated by **hmmpress**.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** option saves output in a simple tabular format that is concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

    -h Help; print a brief reminder of command line usage and all available options.

Options for Controlling Output

    -o *<f>* Direct the main human-readable output to a file *<f>* instead of the default stdout.

    --tblout *<f>* Save a simple tabular (space-delimited) file summarizing the per-hit output, with one data line per homologous target model hit found.

    --dfamtblout *<f>* Save a tabular (space-delimited) file summarizing the per-hit output, similar to **--tblout** but more succinct.

    --aliscoresout *<f>* Save to file a list of per-position scores for each hit. This is useful, for example, in identifying regions of high score density for use in resolving overlapping hits from different models.

    --acc Use accessions instead of names in the main output, where available for profiles and/or sequences.

    --noali Omit the alignment section from the main output. This can greatly reduce the output volume.

--notextw Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

--textw $<n>$ Set the main output's line length limit to $<n>$ characters per line. The default is 120.

Options for Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--dfamtblout*). Hits are ranked by statistical significance (E-value).

-E $<x>$ Report target profiles with an E-value of $<= <x>$. The default is 10.0, meaning that on average, about 10 false positives will be reported per query, so you can see the top of the noise and decide for yourself if it's really noise.

-T $<x>$ Instead of thresholding output on E-value, instead report target profiles with a bit score of $>= <x>$.

Options for Inclusion Thresholds

Inclusion thresholds are stricter than reporting thresholds. Inclusion thresholds control which hits are considered to be reliable enough to be included in an output alignment or a subsequent search round. In **nhmmscan**, which does not have any alignment output (like **nhmmer**), inclusion thresholds have little effect. They only affect what hits get marked as significant (!) or questionable (?) in hit output.

--incE $<x>$ Use an E-value of $<= <x>$ as the inclusion threshold. The default is 0.01, meaning that on average, about 1 false positive would be expected in every 100 searches with different query sequences.

--incT $<x>$ Instead of using E-values for setting the inclusion threshold, use a bit score of $>= <x>$ as the inclusion threshold. It would be unusual to use bit score thresholds with *hmmscan*, because you don't expect a single score threshold to work for different profiles; different profiles have slightly different expected score distributions.

Options for Model-specific Score Thresholding

Curated profile databases may define specific bit score thresholds for each profile, superseding any thresholding based on statistical significance alone. To use these options, the profile must contain the appropriate (GA, TC, and/or NC) optional score threshold annotation; this is picked up by **hmmbuild** from Stockholm format alignment files. For a nucleotide model, each thresholding option has a single per-hit threshold $<x>$ This acts as if **-T**$<x>$ **--incT**$<x>$ has been applied specifically using each model's curated thresholds.

--cut_ga Use the GA (gathering) bit score threshold in the model to set per-hit reporting and inclusion thresholds. GA thresholds are generally considered to be the reliable curated thresholds defining family membership; for example, in Dfam, these thresholds are applied when annotating a genome with a model of a family known to be found in that organism. They may allow for minimal expected false discovery rate.

--cut_nc Use the NC (noise cutoff) bit score threshold in the model to set per-hit reporting and inclusion thresholds. NC thresholds are less stringent than GA; in the context of Pfam, they are generally used to store the score of the highest-scoring known false positive.

--cut_tc Use the NC (trusted cutoff) bit score threshold in the model to set per-hit reporting and inclusion thresholds. TC thresholds are more stringent than GA, and are generally considered to be the score of the lowest-scoring known true positive that is above all known false positives; for example, in Dfam, these thresholds are applied when annotating a genome with a model of a family not known to be found in that organism.

Control of the Acceleration Pipeline

HMMER3 searches are accelerated in a three-step filter pipeline: the scanning-SSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm. There is also a bias filter step between SSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max Turn off (nearly) all filters, including the bias filter, and run full Forward/Backward postprocessing on most of the target sequence. In contrast to **hmmscan,** where this flag really does turn off the filters entirely, the *--max* flag in **nhmmscan** sets the scanning-SSV filter threshold to 0.4, not 1.0. Use of this flag increases sensitivity somewhat, at a large cost in speed.

--F1 $<x>$ Set the P-value threshold for the MSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Other Options

--nonull2 Turn off the null2 score corrections for biased composition.

-Z $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-sequence E-value calculations, rather than the actual number of targets seen.

--seed *<n>*   Set the random number seed to *<n>*. Some steps in postprocessing require Monte Carlo simulation. The default is to use a fixed seed (42), so that results are exactly reproducible. Any other positive integer will give different (but also reproducible) results. A choice of 0 uses an arbitrarily chosen seed.

--qformat *<s>*   Assert that the query sequence file is in format *<s>*. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*. The default is to autodetect the format of the file.

--w_beta *<x>*   Window length tail mass. The upper bound, W, on the length at which nhmmer expects to find an instance of the model is set such that the fraction of all sequences generated by the model with length $>= W$ is less than *<x>*. The default is 1e-7. This flag may be used to override the value of W established for the model by **hmmbuild**.

--w_length *<n>*   Override the model instance length upper bound, W, which is otherwise controlled by **--w_beta**. It should be larger than the model length. The value of W is used deep in the acceleration pipeline, and modest changes are not expected to impact results (though larger values of W do lead to longer run time). This flag may be used to override the value of W established for the model by **hmmbuild**.

--toponly   Only search the top strand. By default both the query sequence and its reverse-complement are searched.

--bottomonly   Only search the bottom (reverse-complement) strand. By default both the query sequence and its reverse-complement are searched.

--cpu *<n>*   Set the number of parallel worker threads to *<n>*. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting *<n>* higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off for your site or machine for some reason.

--stall   For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--mpi   Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

`phmmer` - search protein sequence(s) against a protein sequence database

Synopsis

phmmer *[options]* <*seqfile*> <*seqdb*>

Description

phmmer is used to search one or more query protein sequences against a protein sequence database. For each query sequence in <*seqfile*>, use that sequence to search the target database of sequences in <*seqdb*>, and output ranked lists of the sequences with the most significant matches to the query.

Either the query <*seqfile*> or the target <*seqdb*> may be '-' (a dash character), in which case the query sequences or target database input will be read from a <stdin> pipe instead of from a file. Only one input source can come through <stdin>, not both. An exception is that if the <*seqfile*> contains more than one query sequence, then <*seqdb*> cannot come from <stdin>, because we can't rewind the streaming target database to search it with another query.

The output format is designed to be human-readable, but is often so voluminous that reading it is impractical, and parsing it is a pain. The **--tblout** and **--domtblout** options save output in simple tabular formats that are concise and easier to parse. The **-o** option allows redirecting the main output, including throwing it away in /dev/null.

Options

    -h Help; print a brief reminder of command line usage and all available options.

Options for Controlling Output

    -o <*f*> Direct the main human-readable output to a file <*f*> instead of the default stdout.

    -A <*f*> Save a multiple alignment of all significant hits (those satisfying inclusion thresholds) to the file <*f*> in Stockholm format.

    --tblout <*f*> Save a simple tabular (space-delimited) file summarizing the per-target output, with one data line per homologous target sequence found.

    --domtblout <*f*> Save a simple tabular (space-delimited) file summarizing the per-domain output, with one data line per homologous domain detected in a query sequence for each homologous model.

    --acc Use accessions instead of names in the main output, where available for profiles and/or sequences.

    --noali Omit the alignment section from the main output. This can greatly reduce the output volume.

    --notextw Unlimit the length of each line in the main output. The default is a limit of 120 characters per line, which helps in displaying the output cleanly on terminals and in editors, but can truncate target profile description lines.

    --textw <*n*> Set the main output's line length limit to <*n*> characters per line. The default is 120.

Options Controlling Scoring System

The probability model in **phmmer** is constructed by inferring residue probabilities from a standard 20x20 substitution score matrix, plus two additional parameters for position-independent gap open and gap extend probabilities.

--popen $<x>$ Set the gap open probability for a single sequence query model to $<x>$. The default is 0.02. $<x>$ must be $>= 0$ and < 0.5.

--pextend $<x>$ Set the gap extend probability for a single sequence query model to $<x>$. The default is 0.4. $<x>$ must be $>= 0$ and < 1.0.

--mx $<s>$ Obtain residue alignment probabilities from the built-in substitution matrix named $<s>$. Several standard matrices are built-in, and do not need to be read from files. The matrix name $<s>$ can be PAM30, PAM70, PAM120, PAM240, BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, or BLOSUM90. Only one of the **--mx** and **--mxfile** options may be used.

--mxfile $<mxfile>$ Obtain residue alignment probabilities from the substitution matrix in file $<mxfile>$. The default score matrix is BLOSUM62 (this matrix is internal to HMMER and does not have to be available as a file). The format of a substitution matrix $<mxfile>$ is the standard format accepted by BLAST, FASTA, and other sequence analysis software. Only one of the **--mx** and **--mxfile** options may be used.

Options Controlling Reporting Thresholds

Reporting thresholds control which hits are reported in output files (the main output, *--tblout*, and *--domtblout*). Sequence hits and domain hits are ranked by statistical significance (E-value) and output is generated in two sections called per-target and per-domain output. In per-target output, by default, all sequence hits with an E-value $<= 10$ are reported. In the per-domain output, for each target that has passed per-target reporting thresholds, all domains satisfying per-domain reporting thresholds are reported. By default, these are domains with conditional E-values of $<= 10$. The following options allow you to change the default E-value reporting thresholds, or to use bit score thresholds instead.

-E $<x>$ In the per-target output, report target sequences with an E-value of $<= <x>$. The default is 10.0, meaning that on average, about 10 false positives will be reported per query, so you can see the top of the noise and decide for yourself if it's really noise.

-T $<x>$ Instead of thresholding per-profile output on E-value, instead report target sequences with a bit score of $>= <x>$.

--domE $<x>$ In the per-domain output, for target sequences that have already satisfied the per-profile reporting threshold, report individual domains with a conditional E-value of $<= <x>$. The default is 10.0. A conditional E-value means the expected number of additional false positive domains in the smaller search space of those comparisons that already satisfied the per-target reporting threshold (and thus must have at least one homologous domain already).

--domT $<x>$ Instead of thresholding per-domain output on E-value, instead report domains with a bit score of $>= <x>$.

Options Controlling Inclusion Thresholds

Inclusion thresholds are stricter than reporting thresholds. They control which hits are included in any output multiple alignment (the -*A* option) and which domains are marked as significant ("!") as opposed to questionable ("?") in domain output.

--incE $<x>$ Use an E-value of $<= <x>$ as the per-target inclusion threshold. The default is 0.01, meaning that on average, about 1 false positive would be expected in every 100 searches with different query sequences.

--incT $<x>$ Instead of using E-values for setting the inclusion threshold, instead use a bit score of $>= <x>$ as the per-target inclusion threshold. By default this option is unset.

--incdomE $<x>$ Use a conditional E-value of $<= <x>$ as the per-domain inclusion threshold, in targets that have already satisfied the overall per-target inclusion threshold. The default is 0.01.

--incdomT $<x>$ Instead of using E-values, use a bit score of $>= <x>$ as the per-domain inclusion threshold. By default this option is unset.

Options Controlling the Acceleration Pipeline

HMMER3 searches are accelerated in a three-step filter pipeline: the MSV filter, the Viterbi filter, and the Forward filter. The first filter is the fastest and most approximate; the last is the full Forward scoring algorithm, slowest but most accurate. There is also a bias filter step between MSV and Viterbi. Targets that pass all the steps in the acceleration pipeline are then subjected to postprocessing -- domain identification and scoring using the Forward/Backward algorithm. Essentially the only free parameters that control HMMER's heuristic filters are the P-value thresholds controlling the expected fraction of nonhomologous sequences that pass the filters. Setting the default thresholds higher will pass a higher proportion of nonhomologous sequence, increasing sensitivity at the expense of speed; conversely, setting lower P-value thresholds will pass a smaller proportion, decreasing sensitivity and increasing speed. Setting a filter's P-value threshold to 1.0 means it will passing all sequences, and effectively disables the filter. Changing filter thresholds only removes or includes targets from consideration; changing filter thresholds does not alter bit scores, E-values, or alignments, all of which are determined solely in postprocessing.

--max Maximum sensitivity. Turn off all filters, including the bias filter, and run full Forward/Backward postprocessing on every target. This increases sensitivity slightly, at a large cost in speed.

--F1 $<x>$ First filter threshold; set the P-value threshold for the MSV filter step. The default is 0.02, meaning that roughly 2% of the highest scoring nonhomologous targets are expected to pass the filter.

--F2 $<x>$ Second filter threshold; set the P-value threshold for the Viterbi filter step. The default is 0.001.

--F3 $<x>$ Third filter threshold; set the P-value threshold for the Forward filter step. The default is 1e-5.

--nobias Turn off the bias filter. This increases sensitivity somewhat, but can come at a high cost in speed, especially if the query has biased residue composition (such as a repetitive sequence region, or if it is a membrane protein with large regions of hydrophobicity). Without the bias filter, too many sequences may pass the filter

with biased queries, leading to slower than expected performance as the computationally intensive Forward/Backward algorithms shoulder an abnormally heavy load.

Options Controlling E-value Calibration

Estimating the location parameters for the expected score distributions for MSV filter scores, Viterbi filter scores, and Forward scores requires three short random sequence simulations.

--EmL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EmN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for MSV filter E-values. Default is 200.

--EvL $<n>$ Sets the sequence length in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EvN $<n>$ Sets the number of sequences in simulation that estimates the location parameter mu for Viterbi filter E-values. Default is 200.

--EfL $<n>$ Sets the sequence length in simulation that estimates the location parameter tau for Forward E-values. Default is 100.

--EfN $<n>$ Sets the number of sequences in simulation that estimates the location parameter tau for Forward E-values. Default is 200.

--Eft $<x>$ Sets the tail mass fraction to fit in the simulation that estimates the location parameter tau for Forward evalues. Default is 0.04.

Other Options

--nonull2   Turn off the null2 score corrections for biased composition.

-Z $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-sequence E-value calculations, rather than the actual number of targets seen.

--domZ $<x>$ Assert that the total number of targets in your searches is $<x>$, for the purposes of per-domain conditional E-value calculations, rather than the number of targets that passed the reporting thresholds.

--seed $<n>$ Seed the random number generator with $<n>$, an integer $>= 0$. If $<n>$ is >0, any stochastic simulations will be reproducible; the same command will give the same results. If $<n>$ is 0, the random number generator is seeded arbitrarily, and stochastic simulations will vary from run to run of the same command. The default seed is 42.

--qformat $<s>$ Declare that the input $<seqfile>$ is in format $<s>$. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*. The default is to autodetect the format of the file.

--tformat $<s>$ Declare that the input $<seqdb>$ is in format $<s>$. Accepted formats include *fasta*, *embl*, *genbank*, *ddbj*, *uniprot*, *stockholm*, *pfam*, *a2m*, and *afa*. The default is to autodetect the format of the file.

--cpu *<n>* Set the number of parallel worker threads to *<n>*. By default, HMMER sets this to the number of CPU cores it detects in your machine - that is, it tries to maximize the use of your available processor cores. Setting *<n>* higher than the number of available cores is of little if any value, but you may want to set it to something less. You can also control this number by setting an environment variable, *HMMER_NCPU*. This option is only available if HMMER was compiled with POSIX threads support. This is the default, but it may have been turned off at compile-time for your site or machine for some reason.

--stall For debugging the MPI master/worker version: pause after start, to enable the developer to attach debuggers to the running master and worker(s) processes. Send SIGCONT signal to release the pause. (Under gdb: *(gdb) signal SIGCONT*) (Only available if optional MPI support was enabled at compile-time.)

--mpi Run in MPI master/worker mode, using *mpirun*. (Only available if optional MPI support was enabled at compile-time.)

8 File formats

HMMER profile HMM files

The file `tutorial/fn3.hmm` gives an example of a HMMER3 ASCII save file. An ==abridged== version is shown here, where (...) mark deletions made for clarity and space:

```
HMMER3/f [3.1 | February 2013]
NAME  fn3
ACC   PF00041.13
DESC  Fibronectin type III domain
LENG  86
ALPH  amino
RF    no
MM    no
CONS  yes
CS    yes
MAP   yes
DATE  Fri Feb 15 06:04:13 2013
NSEQ  106
EFFN  11.415833
CKSUM 3564431818
GA    8.00 7.20
TC    8.00 7.20
NC    7.90 7.90
STATS LOCAL MSV      -9.4043  0.71847
STATS LOCAL VITERBI  -9.7737  0.71847
STATS LOCAL FORWARD  -3.8341  0.71847
HMM         A        C        D        E        F        G        H        I   (...)    Y
          m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO  2.70330  4.91262  3.03272  2.64079  3.60307  2.84344  3.74204  3.07942 (...) 3.21526
         2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354 (...) 3.61503
         0.00338  6.08833  6.81068  0.61958  0.77255  0.00000        *
      1  3.16986  5.21447  4.52134  3.29953  4.34285  4.18764  4.30886  3.35801 (...) 3.93889      1 p - - -
         2.68629  4.42236  2.77530  2.73088  3.46365  2.40512  3.72505  3.29365 (...) 3.61514
         0.09796  2.38361  6.81068  0.10064  2.34607  0.48576  0.95510
      2  2.70230  5.97353  2.24744  2.62947  5.31433  2.60356  4.43584  4.79731 (...) 4.25623      3 s - - -
         2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354 (...) 3.61503
         0.00338  6.08833  6.81068  0.61958  0.77255  0.48576  0.95510
(...)
     85  2.48488  5.72055  3.87501  1.97538  3.04853  3.48010  4.51877  3.51898 (...) 3.43366    120 e - - B

         2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354 (...) 3.61503
         0.00338  6.08833  6.81068  0.61958  0.77255  0.48576  0.95510
     86  3.03720  5.94099  3.75455  2.96917  5.26587  2.91682  3.66571  4.11840 (...) 4.99111    121 s - - E

         2.68618  4.42225  2.77519  2.73123  3.46354  2.40513  3.72494  3.29354 (...) 3.61503
         0.00227  6.08723        *  0.61958  0.77255  0.00000        *
//
```

An HMM file consists of one or more HMMs. Each HMM starts with a format version identifier (here, `HMMER3/f`) and ends with `//` on a line by itself. The format version identifier allows backward compatibility as the HMMER software evolves: it tells the parser this file is from HMMER3's save file format version f.[*] The closing `//` allows multiple HMMs to be concatenated.

The format is divided into two regions. The first region contains textual information and miscelleneous parameters in a roughly tag-value scheme. This section ends with a line beginning with the keyword `HMM`. The second region is a tabular, whitespace-limited format for the main model parameters.

All probability parameters are all stored as negative natural log probabilities with five digits of precision to the right of the decimal point, rounded. For example, a probability of 0.25 is stored as $-\log 0.25 = 1.38629$. The special case of a zero probability is stored as '*'.

Spacing is arranged for human readability, but the parser only cares that fields are separated by at least one space character.

A more detailed description of the format follows.

header section

The header section is parsed line by line in a tag/value format. Each line type is either **mandatory** or **optional** as indicated.

HMMER3/f Unique identifier for the save file format version; the `/f` means that this is HMMER3 HMM file format version f. When HMMER3 changes its save file format, the revision code advances. This way,

---

[*]HMMER 3.0 used 3/b format. HMMER 3.1 uses 3/f format. Some alpha test versions of 3.0 used 3/a format. Internal development versions of 3.1 used 3/c, 3/d, and 3/e formats.

parsers may easily remain backwards compatible. The remainder of the line after the `HMMER3/f` tag is free text that is ignored by the parser. HMMER currently writes its version number and release date in brackets here, e.g. `[3.1b2 | December 2013]` in this example. **Mandatory.**

NAME `<s>` Model name; `<s>` is a single word containing no spaces or tabs. The name is normally picked up from the `#=GF ID` line from a Stockholm alignment file. If this is not present, the name is created from the name of the alignment file by removing any file type suffix. For example, an otherwise nameless HMM built from the alignment file `rrm.slx` would be named `rrm`. **Mandatory.**

ACC `<s>` Accession number; `<s>` is a one-word accession number. This is picked up from the `#=GF AC` line in a Stockholm format alignment. **Optional.**

DESC `<s>` Description line; `<s>` is a one-line free text description. This is picked up from the `#=GF DE` line in a Stockholm alignment file. **Optional.**

LENG `<d>` Model length; `<d>`, a positive nonzero integer, is the number of match states in the model. **Mandatory.**

MAXL `<d>` Max instance length; `<d>`, a positive nonzero integer, is the upper bound on the length at which and instance of the model is expected to be found. Used only by nhmmer and nhmmscan. **Optional.**

ALPH `<s>` Symbol alphabet type. For biosequence analysis models, `<s>` is `amino`, `DNA`, or `RNA` (case insensitive). There are also other accepted alphabets for purposes beyond biosequence analysis, including `coins`, `dice`, and `custom`. This determines the symbol alphabet and the size of the symbol emission probability distributions. If `amino`, the alphabet size K is set to 20 and the symbol alphabet to "ACDEFGHIKLMNPQRSTVWY" (alphabetic order); if `DNA`, the alphabet size K is set to 4 and the symbol alphabet to "ACGT"; if `RNA`, the alphabet size K is set to 4 and the symbol alphabet to "ACGU". **Mandatory.**

RF `<s>` Reference annotation flag; `<s>` is either `no` or `yes` (case insensitive). If `yes`, the reference annotation character field for each match state in the main model (see below) is valid; if `no`, these characters are ignored. Reference column annotation is picked up from a Stockholm alignment file's `#=GC RF` line. It is propagated to alignment outputs, and also may optionally be used to define consensus match columns in profile HMM construction. **Optional**; assumed to be no if not present.

MM `<s>` Model masked flag; `<s>` is either `no` or `yes` (case insensitive). If `yes`, the model mask annotation character field for each match state in the main model (see below) is valid; if `no`, these characters are ignored. Indicates that the profile model was created such that emission probabilities at masked positions are set to match the background frequency, rather than being set based on observed frequencies in the alignment. Position-specific insertion and deletion rates are not altered, even in masked regions. **Optional**; assumed to be no if not present.

CONS `<s>` Consensus residue annotation flag; `<s>` is either `no` or `yes` (case insensitive). If `yes`, the consensus residue field for each match state in the main model (see below) is valid. If `no`, these characters are ignored. Consensus residue annotation is determined when models are built. For models of single sequences, the consensus is the same as the query sequence. For models of multiple alignments, the consensus is the maximum likelihood residue at each position. Upper case indicates that the model's emission probability for the consensus residue is $\geq$ an arbitrary threshold (0.5 for protein models, 0.9 for DNA/RNA models).

CS `<s>` Consensus structure annotation flag; `<s>` is either `no` or `yes` (case insensitive). If `yes`, the consensus structure character field for each match state in the main model (see below) is valid; if `no` these characters are ignored. Consensus structure annotation is picked up from a Stockholm file's

#=GC SS_cons line, and propagated to alignment displays. **Optional**; assumed to be no if not present.

MAP <s> Map annotation flag; `<s>` is either `no` or `yes` (case insensitive). If set to `yes`, the map annotation field in the main model (see below) is valid; if `no`, that field will be ignored. The HMM/alignment map annotates each match state with the index of the alignment column from which it came. It can be used for quickly mapping any subsequent HMM alignment back to the original multiple alignment, via the model. **Optional**; assumed to be no if not present.

DATE <s> Date the model was constructed; `<s>` is a free text date string. This field is only used for logging purposes.[†] **Optional.**

COM [<n>] <s> Command line log; `<n>` counts command line numbers, and `<s>` is a one-line command. There may be more than one `COM` line per save file, each numbered starting from $n = 1$. These lines record every HMMER command that modified the save file. This helps us reproducibly and automatically log how Pfam models have been constructed, for example. **Optional.**

NSEQ <d> Sequence number; `<d>` is a nonzero positive integer, the number of sequences that the HMM was trained on. This field is only used for logging purposes. **Optional.**

EFFN <f> Effective sequence number; `<f>` is a nonzero positive real, the effective total number of sequences determined by `hmmbuild` during sequence weighting, for combining observed counts with Dirichlet prior information in parameterizing the model. This field is only used for logging purposes. **Optional.**

CKSUM <d> Training alignment checksum; `<d>` is a nonnegative unsigned 32-bit integer. This number is calculated from the training sequence data, and used in conjunction with the alignment map information to verify that a given alignment is indeed the alignment that the map is for. **Optional.**

GA <f> <f> Pfam gathering thresholds GA1 and GA2. See Pfam documentation of GA lines. **Optional.**

TC <f> <f> Pfam trusted cutoffs TC1 and TC2. See Pfam documentation of TC lines. **Optional.**

NC <f> <f> Pfam noise cutoffs NC1 and NC2. See Pfam documentation of NC lines. **Optional.**

STATS <s1> <s2> <f1> <f2> Statistical parameters needed for E-value calculations. `<s1>` is the model's alignment mode configuration: currently only `LOCAL` is recognized. `<s2>` is the name of the score distribution: currently `MSV`, `VITERBI`, and `FORWARD` are recognized. `<f1>` and `<f2>` are two real-valued parameters controlling location and slope of each distribution, respectively; μ and λ for Gumbel distributions for MSV and Viterbi scores, and τ and λ for exponential tails for Forward scores. λ values must be positive. All three lines or none of them must be present: when all three are present, the model is considered to be calibrated for E-value statistics. **Optional.**

HMM Flags the start of the main model section. Solely for human readability of the tabular model data, the symbol alphabet is shown on the `HMM` line, aligned to the fields of the match and insert symbol emission distributions in the main model below. The next line is also for human readability, providing column headers for the state transition probability fields in the main model section that follows. Though unparsed after the `HMM` tag, the presence of two header lines is **mandatory:** the parser always skips the line after the `HMM` tag line.

COMPO <f>*K The first line in the main model section may be an optional line starting with **COMPO**: these are the model's overall average match state emission probabilities, which are used as a background residue composition in the "filter null" model. The K fields on this line are log probabilities for each residue in the appropriate biosequence alphabet's order. **Optional.**

[†]HMMER does not use dates for any purpose other than human-readable annotation, so it is no more prone than you are to Y2K, Y2038, or any other date-related eschatology.

main model section

All the remaining fields are **mandatory**.

The first two lines in the main model section are atypical.[‡] They contain information for the core model's BEGIN node. This is stored as model node 0, and match state 0 is treated as the BEGIN state. The begin state is mute, so there are no match emission probabilities. The first line is the insert 0 emissions. The second line contains the transitions from the begin state and insert state 0. These seven numbers are: $B \to M_1$, $B \to I_0$, $B \to D_1$; $I_0 \to M_1$, $I_0 \to I_0$; then a 0.0 and a '*', because by convention, nonexistent transitions from the nonexistent delete state 0 are set to $\log 1 = 0$ and $\log 0 = -\infty = $ '*'.

The remainder of the model has three lines per node, for M nodes (where M is the number of match states, as given by the LENG line). These three lines are (K is the alphabet size in residues):

Match emission line The first field is the node number $(1 \ldots M)$. The parser verifies this number as a consistency check (it expects the nodes to come in order). The next K numbers for match emissions, one per symbol, in alphabetic order.

The next field is the MAP annotation for this node. If MAP was yes in the header, then this is an integer, representing the alignment column index for this match state (1..alen); otherwise, this field is '-'.

The next field is the CONS consensus residue for this node. If CONS was yes in the header, then this is a single character, representing the consensus residue annotation for this match state; otherwise, this field is '-'.

The next field is the RF annotation for this node. If RF was yes in the header, then this is a single character, representing the reference annotation for this match state; otherwise, this field is '-'.

The next field is the MM mask value for this node. If MM was yes in the header, then this is a single 'm' character, indicating that the position was identified as a masked position during model construction; otherwise, this field is '-'.

The next field is the CS annotation for this node. If CS was yes, then this is a single character, representing the consensus structure at this match state; otherwise this field is '-'.

Insert emission line The K fields on this line are the insert emission scores, one per symbol, in alphabetic order.

State transition line The seven fields on this line are the transitions for node k, in the order shown by the transition header line: $M_k \to M_{k+1}, I_k, D_{k+1}$; $I_k \to M_{k+1}, I_k$; $D_k \to M_{k+1}, D_{k+1}$.

For transitions from the final node M, match state $M + 1$ is interpreted as the END state E, and there is no delete state $M + 1$; therefore the final $M_k \to D_{k+1}$ and $D_k \to D_{k+1}$ transitions are always * (zero probability), and the final $D_k \to M_{k+1}$ transition is always 0.0 (probability 1.0).

Finally, the last line of the format is the "//" record separator.

Stockholm, the recommended multiple sequence alignment format

The Pfam and Rfam Consortiums have developed a multiple sequence alignment format called "Stockholm format" that allows rich and extensible annotation.

Most popular multiple alignment file formats can be changed into a minimal Stockholm format file just by adding a Stockholm header line and a trailing // terminator:

[‡]That is, the first two lines after the optional COMPO line. Don't be confused by the presence of an optional COMPO line here. The COMPO line is placed in the model section, below the residue column headers, because it's an array of numbers much like residue scores, but it's not really part of the model.

```
# STOCKHOLM 1.0

seq1  ACDEF...GHIKL
seq2  ACDEF...GHIKL
seq3  ...EFMNRGHIKL

seq1  MNPQTVWY
seq2  MNPQTVWY
seq3  MNPQT...
//
```

The first line in the file must be `# STOCKHOLM 1.x`, where `x` is a minor version number for the format specification (and which currently has no effect on my parsers). This line allows a parser to instantly identify the file format.

In the alignment, each line contains a name, followed by the aligned sequence. A dash, period, underscore, or tilde (but not whitespace) denotes a gap. If the alignment is too long to fit on one line, the alignment may be split into multiple blocks, with blocks separated by blank lines. The number of sequences, their order, and their names must be the same in every block. Within a given block, each (sub)sequence (and any associated `#=GR` and `#=GC` markup, see below) is of equal length, called the *block length*. Block lengths may differ from block to block. The block length must be at least one residue, and there is no maximum.

Other blank lines are ignored. You can add comments anywhere to the file (even within a block) on lines starting with a `#`.

All other annotation is added using a tag/value comment style. The tag/value format is inherently extensible, and readily made backwards-compatible; unrecognized tags will simply be ignored. Extra annotation includes consensus and individual RNA or protein secondary structure, sequence weights, a reference coordinate system for the columns, and database source information including name, accession number, and coordinates (for subsequences extracted from a longer source sequence) See below for details.

syntax of Stockholm markup

There are four types of Stockholm markup annotation, for per-file, per-sequence, per-column, and per-residue annotation:

`#=GF <tag> <s>` Per-file annotation. `<s>` is a free format text line of annotation type `<tag>`. For example, `#=GF DATE April 1, 2000`. Can occur anywhere in the file, but usually all the `#=GF` markups occur in a header.

`#=GS <seqname> <tag> <s>` Per-sequence annotation. `<s>` is a free format text line of annotation type `tag` associated with the sequence named `<seqname>`. For example, `#=GS seq1 SPECIES_SOURCE Caenorhabditis elegans`. Can occur anywhere in the file, but in single-block formats (e.g. the Pfam distribution) will typically follow on the line after the sequence itself, and in multi-block formats (e.g. HMMER output), will typically occur in the header preceding the alignment but following the `#=GF` annotation.

`#=GC <tag> <..s..>` Per-column annotation. `<..s..>` is an aligned text line of annotation type `<tag>`. `#=GC` lines are associated with a sequence alignment block; `<..s..>` is aligned to the residues in the alignment block, and has the same length as the rest of the block. Typically `#=GC` lines are placed at the end of each block.

`#=GR <seqname> <tag> <..s..>` Per-residue annotation. `<..s..>` is an aligned text line of annotation type `<tag>`, associated with the sequence named `<seqname>`. `#=GR` lines are associated with one sequence in a sequence alignment block; `<..s..>` is aligned to the residues in that sequence, and has the same

length as the rest of the block. Typically `#=GR` lines are placed immediately following the aligned sequence they annotate.

semantics of Stockholm markup

Any Stockholm parser will accept syntactically correct files, but is not obligated to do anything with the markup lines. It is up to the application whether it will attempt to interpret the meaning (the semantics) of the markup in a useful way. At the two extremes are the Belvu alignment viewer and the HMMER profile hidden Markov model software package.

Belvu simply reads Stockholm markup and displays it, without trying to interpret it at all. The tag types (`#=GF`, etc.) are sufficient to tell Belvu how to display the markup: whether it is attached to the whole file, sequences, columns, or residues.

HMMER uses Stockholm markup to pick up a variety of information from the Pfam multiple alignment database. The Pfam consortium therefore agrees on additional syntax for certain tag types, so HMMER can parse some markups for useful information. This additional syntax is imposed by Pfam, HMMER, and other software of mine, not by Stockholm format per se. You can think of Stockholm as akin to XML, and what my software reads as akin to an XML DTD, if you're into that sort of structured data format lingo.

The Stockholm markup tags that are parsed semantically by my software are as follows:

recognized #=GF annotations

ID <s> Identifier. **<s>** is a name for the alignment; e.g. "rrm". One word. Unique in file.

AC <s> Accession. **<s>** is a unique accession number for the alignment; e.g. "PF00001". Used by the Pfam database, for instance. Often a alphabetical prefix indicating the database (e.g. "PF") followed by a unique numerical accession. One word. Unique in file.

DE <s> Description. **<s>** is a free format line giving a description of the alignment; e.g. "RNA recognition motif proteins". One line. Unique in file.

AU <s> Author. **<s>** is a free format line listing the authors responsible for an alignment; e.g. "Bateman A". One line. Unique in file.

GA <f> <f> Gathering thresholds. Two real numbers giving HMMER bit score per-sequence and per-domain cutoffs used in gathering the members of Pfam full alignments.

NC <f> <f> Noise cutoffs. Two real numbers giving HMMER bit score per-sequence and per-domain cutoffs, set according to the highest scores seen for unrelated sequences when gathering members of Pfam full alignments.

TC <f> <f> Trusted cutoffs. Two real numbers giving HMMER bit score per-sequence and per-domain cutoffs, set according to the lowest scores seen for true homologous sequences that were above the GA gathering thresholds, when gathering members of Pfam full alignments.

recognized #=GS annotations

WT <f> Weight. **<f>** is a positive real number giving the relative weight for a sequence, usually used to compensate for biased representation by downweighting similar sequences. Usually the weights average 1.0 (e.g. the weights sum to the number of sequences in the alignment) but this is not required. Either every sequence must have a weight annotated, or none of them can.

AC <s> Accession. **<s>** is a database accession number for this sequence. (Compare the `#=GF AC` markup, which gives an accession for the whole alignment.) One word.

DE `<s>` Description. `<s>` is one line giving a description for this sequence. (Compare the `#=GF DE` markup, which gives a description for the whole alignment.)

recognized #=GC annotations

RF Reference line. Any character is accepted as a markup for a column. The intent is to allow labeling the columns with some sort of mark.

MM Model mask line. An 'm' indicates that the column lies within a masked range, so that `hmmbuild` should produce emissions matching the background for a match state corresponding to that alignment column. Otherwise, a '.' is used.

SS_cons Secondary structure consensus. For protein alignments, DSSP codes or gaps are accepted as markup: [HGIEBTSCX.-], where H is alpha helix, G is 3/10-helix, I is p-helix, E is extended strand, B is a residue in an isolated b-bridge, T is a turn, S is a bend, C is a random coil or loop, and X is unknown (for instance, a residue that was not resolved in a crystal structure).

SA_cons Surface accessibility consensus. 0-9, gap symbols, or X are accepted as markup. 0 means <10% accessible residue surface area, 1 means <20%, 9 means <100%, etc. X means unknown structure.

recognized #=GR annotations

SS Secondary structure consensus. See `#=GC SS_cons` above.

SA Surface accessibility consensus. See `#=GC SA_cons` above.

PP Posterior probability for an aligned residue. This represents the probability that this residue is assigned to the HMM state corresponding to this alignment column, as opposed to some other state. (Note that a residue can be confidently *unaligned*: a residue in an insert state or flanking N or C state may have high posterior probability.) The posterior probability is encoded as 11 possible characters `0-9*`: $0.0 \leq p < 0.05$ is coded as 0, $0.05 \leq p < 0.15$ is coded as 1, (... and so on ...), $0.85 \leq p < 0.95$ is coded as 9, and $0.95 \leq p \leq 1.0$ is coded as '*'. Gap characters appear in the PP line where no residue has been assigned.

A2M multiple alignment format

HMMER's Easel library routines are capable of writing alignments in UC Santa Cruz "A2M" (alignment to model) format, the native input format for the UCSC SAM profile HMM software package.

To select A2M format, use the format code `a2m`: for example, to reformat a Stockholm alignment to A2M:

```
> esl-reformat a2m myali.sto.
```

Easel currently does not read A2M format, and it currently only writes in what UCSC calls "dotless" A2M format.

The most official documentation for A2M format appears to be at `http://compbio.soe.ucsc.edu/a2m-desc.html`. You may refer to that document if anything in the brief description below is unclear.

An example A2M file

This alignment:

```
seq1  ACDEF...GHIKLMNPQTVWY
seq2  ACDEF...GHIKLMNPQTVWY
seq3  ---EFmnrGHIKLMNPQT---
```

is encoded in A2M format as:

```
>seq1  Sequence 1 description
ACDEFGHIKLMNPQTVWY
>seq2  Sequence 2 description
ACDEFGHIKLMNPQTVWY
>seq3  Sequence 3 description
---EFmnrGHIKLMNPQT---
```

A2M format looks a lot like aligned FASTA format. A crucial difference is that the aligned sequences in a "dotless" A2M file do not necessarily all have the same number of characters. The format distinguishes between "consensus columns" (where residues are in upper case and gaps are a dash, '-') and "insert columns" (where residues are in lower case and gaps are dots, '.', that aren't explicitly shown in the format – hence "dotless" A2M). The position and number of gaps in insert columns (dots) is implicit in this representation. An advantage of this format is its compactness.

This representation only works if all insertions relative to consensus are considered to be unaligned characters. That is how insertions are handled by profile HMM implementations like SAM and HMMER, and profile SCFG implementations like Infernal.

Thus every sequence must have the same number of consensus columns (upper case letters plus '-' characters), and may have additional lower case letters for insertions.

Legal characters

A2M (and SAM) do not support some special characters such as the '*' (not-a-residue) or '~' (missing data) characters. Easel outputs these characters as gaps: either '-' in a consensus column, or nothing in an insert column.

The SAM software parses only a subset of legal ambiguity codes for amino acids and nucleotides. For amino acids, it only reads {BXZ} in addition to the 20 standard one-letter codes. For nucleotides, it only reads {NRY} in addition to {ACGTU}. With one crucial exception, it treats all other letters as X or N.

The crucial exception is 'O'. SAM reads an 'O' as the position of a "free insertion module" (FIM), a concept specific to SAM-style profile HMMs. This has no impact on nucleic acid sequences, where 'O' is not a legal character. But in amino acid sequences, 'O' means pyrrolysine, one of the unusual genetically-encoded amino acids. This means that A2M format alignments must not contain pyrrolysine residues, lest they be read as FIMs. For this reason, Easel converts 'O' residues to 'X' when it writes an amino acid alignment in A2M format.

Determining consensus columns

Writing A2M format requires knowing which alignment columns are supposed to be considered consensus and which are considered inserts. If the alignment was produced by HMMER or Infernal, then the alignment has so-called "reference annotation" (what appears as a `#=GC RF` line in Stockholm format) marking the consensus columns.

Often, an alignment has no reference annotation; for example, if it has been read from an alignment format that has no reference annotation line (only Stockholm and SELEX formats support reference annotation). In this case, Easel internally generates a "reasonable" guess at consensus columns, using essentially the same procedure that HMMER's `hmmbuild` program uses by default: sequence fragments (sequences <50% of the mean sequence length in the alignment overall) are ignored, and for the remaining sequences, any column containing ≥ 50% residues is considered to be a consensus column.

hmmpgmd sequence database format

The hmmpgmd sequence database format closely resembles the FASTA format, with slight modification to support use within HMMER's `hmmpgmd` daemon.

The `hmmpgmd` program enables search of one or more sequence databases (e.g. NR, SwissProt, UniProt) within a single instance, having loaded a single sequence file into memory. Because the set of sequences found within the various databases may overlap, the hmmpgmd format allows each sequence to be stored once, and includes a small piece of metadata that indicates, for each sequence, the list of source databases in which the sequence is found. When a search is performed in `hmmpgmd`, a single target database is specified, and search is restricted to sequences belonging to that database.

Furthermore, because a single sequence might be found multiple times within a single sequence database, `hmmpgmd` is designed to compute E-values not just on the total number of non-redundant sequences within a database, but on the total number of sequences in the original (possibly redundant) database, provided those redundant counts are given in the hmmpgmd-formatted file.

The `hmmpgmd` file begins with a single line containing various counts describing the contents of the file, of the form

#res_cnt seq_cnt db_cnt cnt_1 fullcnt_1 cnt_2 fullcnt_2 ... date_stamp

Fields in header line

res_cnt Number of residues in the sequence file.

seq_cnt Number of sequences in the sequence file.

db_cnt Number of databases in the sequence file.

cnt_i Of the sequnces in the file, the number that belong to database `i`. Note that if the file contains only a single database, this will match `seq_cnt`.

fullcnt_i For database `i`, the number of sequences that should be used in computing E-values. If redundant sequences were collapsed out of the original database, this may be larger than `cnt_i`.

FASTA-like sequence format

In the main body of the sequence file, database sequences are stored sequentially, with each entry consisting of a one-line FASTA-like header followed by one or more lines containing the amino acid sequence, like

```
>1 100
ACDEFGHIKLMNPQTVWY
>2 010
ACDKLMNPQTVWYEFGHI
>3 111
EFMNRGHIKLMNPQT
```

Note that the per-entry header line is made up of two parts. The first part is a numeric, incrementing sequence identifier (the i'th entry has the identifier `i`). The second part is a bit string indicating database membership. In this example, sequence 1 is found in database 1, sequence 2 is found in database 2, and sequence 3 found in databases 1, 2, and 3. The number of bits in each bit string should match `db_cnt`.

Because `hmmpgmd` accepts only numeric sequence identifiers, it is necessary to keep track of the mapping between each numeric sequence identifier and the corresponding meta data (e.g. name, accession, description) external to the hmmpgmd-format file, and post-process `hmmpgmd` seach results to apply those fields to the target sequence information. Depending on the size of the map list, this might be easily acheived with a simple perl array, or might require a more heavy-duty mapping backend such as mongodb (http://www.mongodb.org).

Creating a file in hmmpgmd format

The HMMER-Easel tool `esl-reformat` is able to convert a file in unaligned fasta format into an hmmpgmd format file, such as

> **esl-reformat --id_map mydb.hmmpgmd.map hmmpgmd mydb.fasta > mydb.hmmpgmd.**

The optional `--id_map` flag captures sequence name and description information into a simple tabular file, to be used for mapping those values back to `hmmpgmd` query hits.

9 Acknowledgements and history

HMMER 1 was developed on slow weekends in the lab at the MRC Laboratory of Molecular Biology, Cambridge UK, while I was a postdoc with Richard Durbin and John Sulston. I thank the Human Frontier Science Program and the National Institutes of Health for their remarkably enlightened support at a time when I was really supposed to be working on the genetics of neural development in *C. elegans*.

HMMER 1.8, the first public release of HMMER, came in April 1995, shortly after I moved to Washington University in St. Louis. A few bugfix releases followed. A number of more serious modifications and improvements went into HMMER 1.9 code, but 1.9 was never released. Some versions of HMMER 1.9 inadvertently escaped St. Louis and make it to some genome centers, but 1.9 was never documented or supported. HMMER 1.9 collapsed under its own weight in 1996.

HMMER 2 was a nearly complete rewrite, based on the new Plan 7 model architecture. Implementation was begun in November 1996. I thank the Washington University Dept. of Genetics, the NIH National Human Genome Research Institute, and Monsanto for their support during this time. Also, I thank the Biochemistry Academic Contacts Committee at Eli Lilly & Co. for a gift that paid for the trusty Linux laptop on which much of HMMER 2 was written. The laptop was indispensable. Far too much of HMMER was written in coffee shops, airport lounges, transoceanic flights, and Graeme Mitchison's kitchen. The source code still contains a disjointed record of where and when various bits were written.

HMMER then settled into a comfortable middle age, like its primary author – still actively maintained, though dramatic changes seemed increasingly unlikely. HMMER 2.1.1 was the stable release for three years, from 1998-2001. HMMER 2.2g was intended to be a beta release, but became the *de facto* stable release for two more years, 2001-2003. The final release of the HMMER2 series, 2.3, was assembled in spring 2003. The last bugfix release, 2.3.2, came out in October 2003.

If the world worked as I hoped and expected, the combination of the 1998 Durbin/Eddy/Krogh/Mitchison book *Biological Sequence Analysis* and the existence of HMMER2 as a widely-used proof of principle *should* have motivated the widespread adoption of probabilistic modeling methods for sequence analysis, particularly database search. We would declare Victory and move on. Richard Durbin moved on to human genomics; Anders Krogh moved on to pioneer a number of other probabilistic approaches for other biological sequence analysis problems; Graeme Mitchison moved on to quantum computing; I moved on to noncoding RNAs.

Yet BLAST continued to be the most widely used search program. HMMs seemed to be widely considered to be a mysterious and orthogonal black box, rather than a natural theoretical basis for important programs like BLAST. The NCBI, in particular, seemed to be slow to adopt or even understand HMM methods. This nagged at me; the revolution was unfinished!

When we moved the lab to Janelia Farm in 2006, I had to make a decision about what we should spend our time on. It had to be something "Janelian": something that I would work on with my own hands; something that would be difficult to accomplish under the usual reward structures of academic science; and something that would make the largest possible impact on science. I decided that we should aim to replace BLAST with an entirely new generation of software. The result is the HMMER3 project.

Thanks

HMMER is increasingly not just my own work, but the work of great people in my lab, including Steve Johnson, Alex Coventry, Dawn Brooks, Sergi Castellano, Michael Farrar, Travis Wheeler, and Elena Rivas. The current HMMER development team at Janelia Farm includes Sergi, Michael, and Travis as well as myself.

I would call the Janelia computing environment world-class except that it's even better than that. That's entirely due to Goran Ceric. HMMER3 testing now spins up thousands of processors at a time, an unearthly amount of computing power.

Over the years, the MRC-LMB computational molecular biology discussion group contributed many ideas to HMMER. In particular, I thank Richard Durbin, Graeme Mitchison, Erik Sonnhammer, Alex Bate-

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.*, 25:3389–3402.

Barton, G. J. (1990). Protein multiple sequence alignment and flexible pattern matching. *Meth. Enzymol.*, 183:403–427.

Bashford, D., Chothia, C., and Lesk, A. M. (1987). Determinants of a protein fold: Unique features of the globin amino acid sequences. *J. Mol. Biol.*, 196:199–216.

Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. *Bull. Math. Biol.*, 51:79–94.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. J. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK.

Eddy, S. R. (1996). Hidden Markov models. *Curr. Opin. Struct. Biol.*, 6:361–365.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, 14:755–763.

Eddy, S. R. (2008). A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Comput. Biol.*, 4:e1000069.

Eddy, S. R. (2009). A new generation of homology search tools based on probabilistic inference. *Genome Inform.*, 23:205–211.

Eddy, S. R. (2011). Accelerated profile HMM searches. *PLoS Comp. Biol.*, 7:e1002195.

Finn, R. D., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J. E., Gavin, O. L., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E. L. L., Eddy, S. R., and Bateman, A. (2010). The Pfam protein families database. *Nucl. Acids Res.*, 38:D211–D222.

Gonzalez, M. W. and Pearson, W. R. (2010). Homologous over-extension: a challenge for iterative similarity searches. *Nucl. Acids Res.*, 38:2177–2189.

Gribskov, M., Luthy, R., and Eisenberg, D. (1990). Profile analysis. *Meth. Enzymol.*, 183:146–159.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987). Profile analysis: Detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355–4358.

Holmes, I. (1998). *Studies in Probabilistic Sequence Alignment and Evolution*. PhD thesis, University of Cambridge.

Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268.

Karlin, S. and Altschul, S. F. (1993). Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. USA*, 90:5873–5877.

Krogh, A. (1998). An introduction to hidden Markov models for biological sequences. In Salzberg, S., Searls, D., and Kasif, S., editors, *Computational Methods in Molecular Biology*, pages 45–63. Elsevier.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531.

Letunic, I., Copley, R. R., Pils, B., Pinkert, S., Schultz, J., and Bork, P. (2006). SMART 5: Domains in the context of genomes and networks. *Nucl. Acids Res.*, 34:D257–D260.

Mulder, N. J., Apweiler, R., Attwood, T. K., Bairoch, A., Barrell, D., Bateman, A., Binns, D., Biswas, M., Bradley, P., Bork, P., Bucher, P., Copley, R. R., Courcelle, E., Das, U., Durbin, R., Falquet, L., Fleischmann, W., Griffiths-Jones, S., Haft, D., Harte, N., Hulo, N., Kahn, D., Kanapin, A., Krestyaninova, M., Lopez, R., Letunic, I., Lonsdale, D., Silventoinen, V., Orchard, S. E., Pagni, M., Peyruc, D., Ponting, C. P., Selengut, J. D., Servant, F., Sigrist, C. J., Vaughan, R., and Zdobnov, E. M. (2003). The InterPro database, 2003 brings increased coverage and new features. *Nucl. Acids Res.*, 31:315–318.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197.

Sonnhammer, E. L. L., Eddy, S. R., and Durbin, R. (1997). Pfam: A comprehensive database of protein families based on seed alignments. *Proteins*, 28:405–420.

Taylor, W. R. (1986). Identification of protein sequence homology by consensus template alignment. *J. Mol. Biol.*, 188:233–258.

Wheeler, T. J., Clements, J., Eddy, S. R., Hubley, R., Jones, T. A., Jurka, J., Smit, A. F. A., and Finn, R. D. (2013). Dfam: a database of repetitive DNA based on profile hidden Markov models. *Nucl. Acids Res.*, 41:D70–D82.