# HMMER User's Guide

Biological sequence analysis using profile hidden Markov models

*http://hmmer.wustl.edu/*
Version 2.1.1; December 1998

Sean Eddy
Dept. of Genetics, Washington University School of Medicine
4566 Scott Ave., St. Louis, MO 63110, USA
*eddy@genetics.wustl.edu*
With contributions by Ewan Birney (*birney@sanger.ac.uk*)

# Contents

# Chapter 1

# Tutorial

I hate reading documentation. I just want examples of how stuff works, just enough to get me started and doing something productive. So, here's a tutorial walk-through of some small projects with HMMER. If you want the introduction, that's the second chapter. The tutorial should be sufficient to get you started on work of your own. You can read the other chapters later if you want.

## 1.1 The programs in HMMER

There are currently nine programs supported in the HMMER 2 package:

**`hmmalign`** Align sequences to an existing model.

**`hmmbuild`** Build a model from a multiple sequence alignment.

**`hmmcalibrate`** Takes an HMM and empirically determines parameters that are used to make searches more sensitive, by calculating more accurate expectation value scores (E-values).

**`hmmconvert`** Convert a model file into different formats, including a compact HMMER 2 binary format, and "best effort" emulation of GCG profiles.

**`hmmemit`** Emit sequences probabilistically from a profile HMM.

**`hmmfetch`** Get a single model from an HMM database.

**`hmmindex`** Index an HMM database.

**`hmmpfam`** Search an HMM database for matches to a query sequence.

**`hmmsearch`** Search a sequence database for matches to an HMM.

HMMER also provides a number of utility programs which are not HMM programs, but may be useful:

**`alistat`** Show some simple statistics about a sequence alignment file.

**`getseq`** Retrieve a (sub-)sequence from a sequence file.

**`seqstat`** Show some simple statistics about a sequence file.

**`sreformat`** Reformat a sequence file into a different format.

The following four programs are not included in the current release, but are planned (i.e. they're vaporware, but it may be useful to you to know that they're on the drawing board:)

**hmmconfig** Alter the search configuration of an existing model.

**hmminfo** Display summary information about the model(s) in a file.

**hmmtrain** Train a model from initially unaligned sequences, producing both a model and a multiple alignment.

**hmmview** Graphical viewer and editor for HMMs.

## 1.2 Files used in the tutorial

The subdirectory `/Demos` 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:

**globins50.msf** An MSF format alignment file of 50 aligned globin sequences.

**globins630.fa** A FASTA format file of 630 unaligned globin sequences.

**fn3.slx** A SELEX format alignment file of fibronectin type III domains.

**rrm.slx** A SELEX format alignment file of RNA recognition motif domains.

**pkinase.slx** A SELEX format alignment file of protein kinase catalytic domains.

**Artemia.fa** A FASTA file of brine shrimp globin, which contains nine tandemly repeated globin domains.

**7LES_DROME** A SWISSPROT file of the *Drosophila* Sevenless sequence, a receptor tyrosine kinase with multiple domains.

Create a new directory that you can work in, and copy all the files in `Demos` there. I'll assume for the following examples that you've installed the HMMER programs in your path; if not, you'll need to give a complete path name to the HMMER programs (e.g. something like `/usr/people/eddy/ hmmer-2.0/hmmbuild` instead of just `hmmbuild`).

## 1.3 Searching a sequence database with a single profile HMM

One common use of HMMER is to search a sequence database for homologues of a protein family of interest. You need a multiple sequence alignment of the sequence family you're interested in. (Profile HMMs can be trained from unaligned sequences; however, this functionality is temporarily withdrawn from HMMER. I recommend CLUSTALW as an excellent, freely available multiple sequence alignment program.)

## HMM construction with `hmmbuild`

Let's assume you have a multiple sequence alignment of a protein domain or protein sequence family. To use HMMER to search for additional remote homologues of the family, you want to first build a profile HMM from the alignment. The following command builds a profile HMM from the alignment of 50 globin sequences in `globins50.msf`:

```
> hmmbuild globin.hmm globins50.msf
hmmbuild - build a hidden Markov model from an alignment
HMMER 2.0 (June 1998)
Copyright (C) 1992-1998 Washington University School of Medicine
HMMER is freely distributed under the GNU General Public License (GPL).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Training alignment:                 globins50.msf
Number of sequences:                50
Search algorithm configuration:     Multiple domain (hmmls)
Model construction strategy:        MAP (gapmax hint: 0.50)
Prior used:                         (default)
Prior strategy:                     Dirichlet
Sequence weighting method:          G/S/C tree weights
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Determining effective sequence number    ... done. [13]
Weighting sequences heuristically        ... done.
Constructing model architecture          ... done.
Converting counts to probabilities       ... done.
Setting model name, etc.                 ... done. [globins50]

Constructed a profile HMM (length 162)
Average score:      283.03 bits
Minimum score:      137.32 bits
Maximum score:      343.50 bits
Std. deviation:      53.21 bits

Finalizing model configuration           ... done.
Saving model to file                     ... done. [globin.hmm]
```

The process takes a second or two. `hmmbuild` create a new HMM file called `globin.hmm`. This is a human and computer readable ASCII text file, but for now you don't care. You also don't care for now what all the stuff in the output means; I'll describe it in detail later. The profile HMM can be treated as a compiled model of your alignment.

## HMM calibration with `hmmcalibrate`

This step is optional, but doing it will increase the sensitivity of your database search.

When you search a sequence database, it is useful to get "E-values" (expectation values) in addition to raw scores. When you see a database hit that scores $x$, an E-value tells you the number of hits you would've expected to score $x$ or more just by chance in a sequence database of this size.

HMMER will always estimate an E-value for your hits. However, unless you "calibrate" your model before a database search, HMMER uses an analytic upper bound calculation that is extremely conservative. An empirical HMM calibration costs time (about 10% the time of a SWISSPROT search) but it only has to be done once per model, and can greatly increase the sensitivity of a database search. To empirically calibrate the E-value calculations for the globin model, type:

```
> hmmcalibrate globin.hmm
hmmcalibrate -- calibrate HMM search statistics
HMMER 2.0 (June 1998)
Copyright (C) 1992-1998 Washington University School of Medicine
HMMER is freely distributed under the GNU General Public License (GPL).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HMM file:                 globin.hmm
Length distribution mean: 325
Length distribution s.d.: 200
Number of samples:        5000
random seed:              895790561
histogram(s) saved to:    [not saved]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

HMM    : globins50
mu     :  -107.928612
lambda :     0.186579
max    :   -63.859001
//
```

This takes several minutes. Go have a cup of coffee. When it is complete, the relevant parameters are added to the HMM file.

Calibrated HMMER E-values tend to be relatively accurate. E-values of 0.1 or less are, in general, very significant hits. Uncalibrated HMMER E-values are also reliable, erring on the cautious side; uncalibrated models may miss remote homologues.

### Sequence database search with **hmmsearch**

As an example of searching for new homologues using a profile HMM, we'll use the globin model to search for globin domains in the example *Artemia* globin sequence in `Artemia.fa`:

```
> hmmsearch globin.hmm Artemia.fa
```

The output comes in several sections, and unlike building and calibrating the HMM (where we treated the HMM as a black box), now you *do* care about what it's saying.

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

```
hmmsearch - search a sequence database with a profile HMM
HMMER 2.0 (June 1998)
Copyright (C) 1992-1998 Washington University School of Medicine
HMMER is freely distributed under the GNU General Public License (GPL).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HMM file:                 globin.hmm [globins50]
Sequence database:        Artemia.fa
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query HMM:  globins50
  [HMM has been calibrated; E-values are empirical estimates]
```

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:

```
Scores for complete sequences (score includes all domains):
Sequence Description                                    Score    E-value  N
-------- -----------                                    -----    ------- ---
S13421   S13421 GLOBIN - BRINE SHRIMP                   335.5     1e-101   8
```

The first field is the name of the target sequence, then followed by the description line for the sequence. The last three fields are the raw score (in units of "bits"), the estimated E-value, and the total number of domains detected in the sequence. By default, every sequence with an E-value over 10.0 is listed in this output.

The second section is the *domain top hits* list. By default, for every sequence with an E-value less than 10, every domain with a non-zero raw score is listed. (Read that carefully. In a later chapter we'll discuss some caveats about how `hmmsearch` identifies domains, and how to control its output in different ways.) Each domain detected in the search is output in a list ranked by E-value:

```
Parsed for domains:
Sequence Domain  seq-f seq-t    hmm-f hmm-t      score  E-value
-------- ------- ----- -----    ----- -----      -----  -------
S13421     6/8     928  1075 ..     1   162 []    65.8  1.5e-20
S13421     2/8     149   288 ..     1   162 []    59.4  1.3e-18
S13421     3/8     303   450 ..     1   162 []    58.4  2.6e-18
S13421     8/8    1238  1390 ..     1   162 []    42.5  1.6e-13
S13421     5/8     771   918 ..     1   162 []    33.8  3.3e-12
S13421     7/8    1085  1234 ..     1   162 []    32.0  4.6e-12
S13421     4/8     454   607 ..     1   162 []    26.1  1.4e-11
S13421     1/8       1   139 [.     1   162 []    21.9    3e-11
```

The first field is the name of the target sequence. The second field is the number of this domain: e.g. "6/8" means the sixth domain of eight total domains detected.

The fields marked "seq-f" and "seq-t" mean "sequence from" and "sequence to": the start and end points of the alignment on the target sequence. After these two fields is a shorthand annotation for whether the alignment is "global" with respect to the sequence or not. A dot (.) means the alignment does not go all the way to the end; a bracket ([ or ]) means it does. Thus, .. means that the alignment is local within the sequence; [. means that the alignment starts at the beginning of the sequence, but doesn't go all the way to its end; .] means the alignment starts somewhere internally and goes all the way to the end; and [] means the alignment includes the entire sequence.

Analogously, the fields marked "hmm-f" and "hmm-t" indicate the start and end points with respect to the consensus coordinates of the model, and the following field is a shorthand for whether the alignment is global with respect to the *model*. Here, for instance, all the globin domains in the *Artemia* sequence are complete matches to the entire globin model – *because, by default,* `hmmbuild` *built the HMM to only look for those kinds of alignments*. We'll discuss later how to modify the profile HMM for other search styles.

The final two fields are the raw score in bits and the estimated E-value, *for the isolated domain*. Because of the method HMMER uses to correct raw scores for biased sequence composition, the raw scores for the domains do not necessarily sum up to the raw score of the sequence.

The next section is the *alignment output*. By default, every domain that appeared in the domain top hits list now appears as a BLAST-like alignment. For example:

```
Alignments of top-scoring domains:
```

```
S13421: domain 6 of 8, from 928 to 1075: score 65.8, E = 1.5e-20
                   *->vilealvnssShLSaeekalVkslWYgKVegnaeeiGaeaLgRlFvv
                      +           LSa e a Vk++W   V+ ++ ++G  ++  lF +
      S13421   928    G----------LSAREVAVVKQTW-NLVKPDLMGVGMRIFKSLFEA 962

                   YPwTqryFphFgdLssldavkgspkvKaHGkKVltalgdavkhLDdtgnl
                   +P  q+ Fp+F+d+ +ld +++ p v +H   V t l++ ++ LD   nl
      S13421   963 FPAYQAVFPKFSDV-PLDKLEDTPAVGKHSISVTTKLDELIQTLDEPANL 1011

                   kgalakLSelHadklrVDPeNFklLghvlvvvLaehfgkdftPevqAAwd
                   +    +L+e H   lrV+   Fk +g+vlv  L   +g  f+  +  +w
      S13421  1012 ALLARQLGEDH-IVLRVNKPMFKSFGKVLVRLLENDLGQRFSSFASRSWH 1060

                   KflagvanaLahKYr<-*
                   K++++++  +++
      S13421  1061 KAYDVIVEYIEEGLQ    1075
```

The top line is the HMM consensus. The amino acid shown for the consensus is the highest probability amino acid at that position according to the HMM (not necessarily the highest *scoring* amino acid, though). Capital letters mean "highly conserved" residues: those with a probability of $> 0.5$ for protein models, or $> 0.9$ for DNA models.

The center line shows letters for "exact" matches to the highest probability residue in the HMM, or a "+" when the match has a positive score and is therefore considered to be "conservative" according to the HMM's view of *this particular position in the model* – not the usual definition of conservative changes in general.

The third line shows the sequence itself, of course.

The next section of the output is the *score histogram*. It shows a histogram with raw score increasing along the Y axis, and the number of sequence hits represented as a bar along the X axis. In our example here, since there's only a single sequence, the histogram is very boring:

```
Histogram of all scores:
score    obs    exp  (one = represents 1 sequences)
-----    ---    ---
  335      1      0|=
```

Notice though that it's a histogram of the whole sequence hits, not the domain hits.

You can ignore the rest of the `hmmsearch` output:

```
% Statistical details of theoretical EVD fit:
              mu =  -107.9286
          lambda =     0.1866
chi-sq statistic =     0.0000
  P(chi-square)  =          0

Whole sequence top hits:
tophits_s report:
     Total hits:           1
     Satisfying E cutoff:  1
     Total memory:         15K

Domain top hits:
tophits_s report:
```

```
Total hits:           8
Satisfying E cutoff:  8
Total memory:         20K
```

This is just some trailing internal info about the search that's useful to me sometimes, but probably not to you.

### Searching major databases like NR or SWISSPROT

HMMER reads all major database formats and does not need any special database indexing. You can search any large sequence database you have installed locally just by giving the full path to the database file, e.g. something like:

> **`hmmsearch globin.hmm /nfs/databases/swiss35/sprot35.dat`**

If you have BLAST installed locally, it's likely that you have a directory (or directories) in which the BLAST databases are kept. These directories are specified in an environment variable called `BLASTDB`. HMMER will read the same environment variable. For example, if you have a BLAST database called `/nfs/databases/blast-db/swiss35`, the following commands will work:

> **`setenv BLASTDB /nfs/databases/blast-db/`**
> **`hmmsearch globin.hmm swiss35`**

Obviously, you'd tend to have the `setenv` command as part of the local configuration of your machine, rather than typing it at the command line.

### Local alignment searches with **`hmmsearch`**

*This is extremely important.* HMMER does not do local (Smith/Waterman) and global (Needleman/Wunsch) style alignments in the same way that most computational biology analysis programs do it. To HMMER, whether local or global alignments are allowed is part of the *model*, rather than being accomplished by running a different *algorithm*. (This will be discussed in greater detail later; it is part of the "Plan7" architecture of the new HMMER2 models.)

Therefore, you need to choose what kind of alignments you want to allow *when you build the model* with `hmmbuild`. By default, `hmmbuild` builds models which allow alignments that are global with respect to the HMM, local with respect to the sequence, and allows multiple domains to hit per sequence. Such models will only find complete domains.

`hmmbuild` provides some standard options for common alignment styles. The following table shows the four alignment styles supported by `hmmbuild`, and also shows the equivalent old HMMER 1.x search program style (to orient experienced HMMER users).

| Command | w.r.t. sequence | w.r.t. HMM | multidomain | HMMER 1 equivalent |
|---|---|---|---|---|
| `hmmbuild` | local | global | yes | hmmls |
| `hmmbuild -f` | local | local | yes | hmmfs |
| `hmmbuild -g` | local | global | no | hmms |
| `hmmbuild -s` | local | local | no | hmmsw |

In brief, if you want maximum sensitivity at the expense of only finding complete domains, use the `hmmbuild` default. If you need to find fragments (local alignments) too, and are willing to give up some sensitivity to see them, use `hmmbuild -f`. If you want the best of both worlds, build two models and search with both of them.

## 1.4 Searching a query sequence against a profile HMM database

A second use of HMMER is to look for known domains in a query sequence, by searching a single sequence against a library of HMMs. (Contrast the previous section, in which we searched a single HMM against a sequence database.) To do this, you need a library of profile HMMs. One such library is our PFAM database (Sonnhammer et al., 1997; Sonnhammer et al., 1998), and you can also create your own.

### Creating your own profile HMM database

HMM databases are simply concatenated single HMM files. You can build them either by invoking the `-A` "append" option of `hmmbuild`, or by concatenating HMM files you've already built. For example, here's two ways to build an HMM database called `myhmms` that contains models of the rrm RNA recognition motif domain, the fn3 fibronectin type III domain, and the pkinase protein kinase catalytic domain:

```
> hmmbuild rrm.hmm rrm.slx
> hmmbuild fn3.hmm fn3.slx
> hmmbuild pkinase.hmm pkinase.slx
> cat rrm.hmm fn3.hmm pkinase.hmm > myhmms
> hmmcalibrate myhmms
```

or:

```
> hmmbuild -A myhmms rrm.slx
> hmmbuild -A myhmms fn3.slx
> hmmbuild -A myhmms pkinase.slx
> hmmcalibrate myhmms
```

Notice that `hmmcalibrate` can be run on HMM databases as well as single HMMs.

### Parsing the domain structure of a sequence with `hmmpfam`

Now that you have a small HMM database called `myhmms`, let's use it to analyze the *Drosophila* Sevenless sequence, `7LES_DROME`:

```
> hmmpfam myhmms 7LES_DROME
```

Like `hmmsearch`, the `hmmpfam` output comes in several sections. The first section is the *header*:

```
hmmpfam - search a single seq against HMM database
HMMER 2.0 (June 1998)
Copyright (C) 1992-1998 Washington University School of Medicine
HMMER is freely distributed under the GNU General Public License (GPL).
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HMM file:                 myhmms
Sequence file:            7LES_DROME
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Query:  7LES_DROME  SEVENLESS PROTEIN (EC 2.7.1.112).
```

The next section is the *sequence family classification* top hits list, ranked by E-value. The scores and E-values here reflect the confidence that this query sequence contains one *or more* domains belonging to a domain family. The fields have the same meaning as in `hmmsearch` output, except that the name and description are for the HMM that's been hit.

```
Scores for sequence family classification (score includes all domains):
Sequence Description                                    Score    E-value  N
-------- -----------                                    -----    ------- ---
pkinase                                                 303.3      3e-87   1
fn3                                                     171.8    1.1e-47   6
```

The next section is the *domain parse* list, ordered by position on the sequence (not by score). Again the fields have the same meaning as in `hmmsearch` output:

```
Parsed for domains:
Sequence Domain  seq-f seq-t    hmm-f hmm-t      score  E-value
-------- ------- ----- -----    ----- -----      -----  -------
fn3        1/6     437   522 ..     1    84 []    48.0  2.1e-10
fn3        2/6     825   914 ..     1    84 []    12.6     0.21
fn3        3/6    1292  1389 ..     1    84 []    15.2     0.13
fn3        4/6    1799  1891 ..     1    84 []    62.4  9.4e-15
fn3        5/6    1899  1978 ..     1    84 []    13.7     0.17
fn3        6/6    1993  2107 ..     1    84 []    18.4    0.067
pkinase    1/1    2209  2483 ..     1   278 []   303.3    3e-87
```

The final output section is the *alignment output*, just like `hmmsearch`:

```
Alignments of top-scoring domains:
fn3: domain 1 of 6, from 437 to 522: score 48.0, E = 2.1e-10
                   *->P.saPtnltvtdvtstsltlsWsppt.gngpitgYevtyRqpkngge
                      P saP   + +++ ++ l ++W p +  ngpi+gY+++  ++++g+
  7LES_DROME   437    PiSAPVIEHLMGLDDSHLAVHWHPGRfTNGPIEGYRLRL-SSSEGNA 482

                   wneltvpgtttsytltgLkPgteYtvrVqAvnggG.GpeS<-*
                   + e+ vp+   sy+++ L++gt+Yt+ +  +n +G+Gp
  7LES_DROME   483 TSEQLVPAGRGSYIFSQLQAGTNYTLALSMINKQGeGPVA    522
...
```

## Downloading the PFAM database

The PFAM database is available from either `http://pfam.wustl.edu/` or `http://www.sanger.ac.uk/Pfam/`. Download instructions are on the Web page. The PFAM HMM library is a single large file, containing several hundred models of known protein domains. Install it in a convenient directory and name it something simple like `pfam`.

HMMER will look for PFAM and other files in a directory (or directories) specified by the `HMMERDB` environment variable. For instance, if you install the PFAM HMM library as `/nfs/databases/hmmer/pfam`, the following commands will search for domains in `7LES_DROME`:

```
> setenv HMMERDB /nfs/databases/hmmer/
> hmmpfam pfam 7LES_DROME
```

## 1.5 Maintaining multiple alignments with hmmalign

Another use of profile HMMs is to create multiple sequence alignments of large numbers of sequences. A profile HMM can be build of a "seed" alignment of a small number of representative sequences, and this profile HMM can be used to efficiently align any number of additional sequences.

This is in fact how the PFAM database is updated as the main SPTREMBL database increases in size. The PFAM seed alignments are (relatively) stable from release to release; PFAM full alignments are created automatically by searching SPTREMBL with the seed model and aligning all the significant hits into a multiple alignment using `hmmalign`.

For example, to align the 630 globin sequences in `globins630.fa` to our globin model `globin.hmm`, and create a new alignment file called `globins630.ali`, we'd do:

```
> hmmalign -o globins630.ali globin.hmm globins630.fa
hmmalign - align sequences to an HMM profile
HMMER 2.0 (June 1998)
Copyright (C) 1992-1998 Washington University School of Medicine
HMMER is freely distributed under the GNU General Public License (GPL).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HMM file:             globin.hmm
Sequence file:        globins630.fa
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Alignment saved in file globins630.ali
```

Using the `-o` option to specify a save file for the final alignment is a good idea; else, the alignment will be displayed on the screen as output (and an alignment of several hundred sequences will give a fairly voluminous output).

# Chapter 2

# Introduction

You discover a new protein sequence family, and carefully construct a multiple sequence alignment. Your family, like most protein families, has a number of strongly (but not absolutely) conserved key residues, 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. You sure wish there was a computer program that did this better.

You get the sequence of another favorite protein and eagerly BLAST it against the NCBI server, and nothing significant shows up. It looks like you'll get no easy information from the sequence.

You BLAST another favorite sequence against the NCBI server; a thousand hits come back, and the top hundred are hypothetical sequences from genome projects. It looks like you'll have to wade through a lot of BLAST output to find the informative hits.

Another sequence comes back a slew of hits to receptor tyrosine kinases, but before you decide to call your sequence an RTK homologue, you recall that RTK's are, like many proteins, composed of multiple functional domains. Is your sequence really an RTK? Or is it a novel sequence that also happens to have a protein kinase catalytic domain, fibronectin type III domain, or immunoglobulin superfamily domain?

These (and other problems in sequence analysis) are the kinds of problems that HMMER tries to address.

## 2.1 Profile HMMs

Profile hidden Markov models (profile HMMs) are statistical models of the primary structure consensus of a sequence family. Anders Krogh, David Haussler, and co-workers at UC Santa Cruz introduced profile HMMs (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, but the Krogh paper had a particularly 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. Since then, several computational biology groups (including ours) have rapidly adopted HMMs as the underlying formalism for sequence profile analysis.

"Profiles" were introduced by Gribskov and colleagues (Gribskov et al., 1987; Gribskov et al., 1990) at about the same time that other groups introduced similar approaches, such as "flexible patterns" (Barton, 1990), and "templates"(Bashford et al., 1987; Taylor, 1986). The term "profile" has stuck.[1] All of these are

[1] There has been some agitation to call all such models "position specific scoring matrices", PSSMs, but certain small nocturnal

more or less statistical descriptions of the consensus of a multiple sequence alignment. They use position-specific scores for amino acids (or nucleotides) and position specific scores 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 can use Bayesian probability theory to guide how all the probability (scoring) parameters should be set. Though this might sound like a purely academic issue, this probabilistic basis lets us do things that the more heuristic methods cannot do easily. For example, an HMM can be trained from unaligned sequences, if a trusted alignment isn't yet known. Another consequence is that HMMs have a consistent theory behind gap and insertion scores.[2] In most details, HMMs are a slight improvement over a carefully constructed profile – but far less skill and manual intervention is necessary to train a good HMM and use it. This allows us to make libraries of hundreds of profile HMMs and apply them on a very large scale to whole-genome or EST sequence analysis. One such database of protein domain models is Pfam (Sonnhammer et al., 1997); the construction and use of Pfam is tightly tied to the HMMER software package.

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.[3] HMMs make poor models of RNAs, for instance, because an HMM cannot describe base pairs. Also, compare protein "threading" methods, which include scoring terms for nearby amino acids in a three-dimensional protein structure.

A general definition of HMMs and an excellent tutorial introduction to their use has been written by Rabiner (Rabiner, 1989). Throughout, I will often use "HMM" to refer to the specific case of profile HMMs as described by Krogh et al. (Krogh et al., 1994). This shorthand usage is for convenience only. For a review of profile HMMs, see (Eddy, 1996), and for a complete book on the subject of probabilistic modeling in computational biology, see (Durbin et al., 1998) .

## 2.2 Primary changes from HMMER 1.x

HMMER 2 is an almost complete rewrite of the original 1992-1996 HMMER code. A list of the major changes follows.

**Plan7** The model architecture is changed. The "Plan 7" architecture[4] allows detailed control over parameters of local, global, and multidomain alignments. In Plan 7, you choose the alignment style when you build the model with `hmmbuild`, not when you search. A single database search program `hmmsearch` replaces the four search programs in HMMER 1.x.

**Pfam support** There is much better support for the Pfam HMM database. HMMs have names; the HMM file format allows libraries of multiple HMMs per file; and there is a search

North American marsupials have a prior claim on the mnemonic.

[2]Surprisingly, this is still controversial. People have been saying "there is no theory for setting gap scores" for so long that many people believe it.

[3]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).

[4]The origin of the codename is obscure, involving both details of the state transition probability distributions in profile HMM architecture and an inordinate fondness for bad science fiction movies. Frighteningly, David Haussler understood the codename immediately.

program `hmmpfam` for searching a single query sequence against an HMM library. HMM databases like Pfam can be indexed for rapid access using `hmmindex`, and single HMMs can be retrieved from an HMM database using `hmmfetch`.

**E-values** HMMER now reports both log-odds scores and a E-value. E-values can detect significant matches even when the log-odds score is negative. A new program `hmmcalibrate` empirically determines the necessary parameters for estimating E-values as best as possible.

**Output** Database searches now sort and postprocess their results for prettier and more useful output. Output styles are deliberately similar to BLAST, to try to simplify the design of postprocessors.

**Save file format** HMMER save files are now in an ASCII format which is easy for humans to read, portable across all architectures without byteswapping issues, and compact.

**Decent defaults** The "best" average behavior of HMMER is now its default behavior. For example, mixture Dirichlet priors are now the default for protein model building. In HMMER 1.x, you had to invoke a number of "experimental" options to get better performance – this led to a bunch of really frustrating benchmarking in the literature, in which people compared fully optimized program X to the default behavior of HMMER 1, which was the 1994 basic Krogh/Haussler behavior rather than its actual power. (However, there's still a lot of bad benchmarking, because people don't understand how to benchmark profile software yet.)

## 2.3 Plan 7

### The Plan 7 architecture

The Plan 7 architecture is substantially different from the original HMMER model architecture.

The abbreviations for the states are as follows:

**$M_x$** Match state $x$. Has $K$ emission probabilities.

**$D_x$** Delete state $x$. Non-emitter.

**$I_x$** Insert state $x$. Has $K$ emission probabilities.

**S** Start state. Non-emitter.

**N** N-terminal unaligned sequence state. Emits *on transition* with $K$ emission probabilities.

**B** Begin state (for entering main model). Non-emitter.

**E** End state (for exiting main model). Non-emitter.

**C** C-terminal unaligned sequence state. Emits *on transition* with $K$ emission probabilities.

**J** Joining segment unaligned sequence state. Emits *on transition* with $K$ emission probabilities.

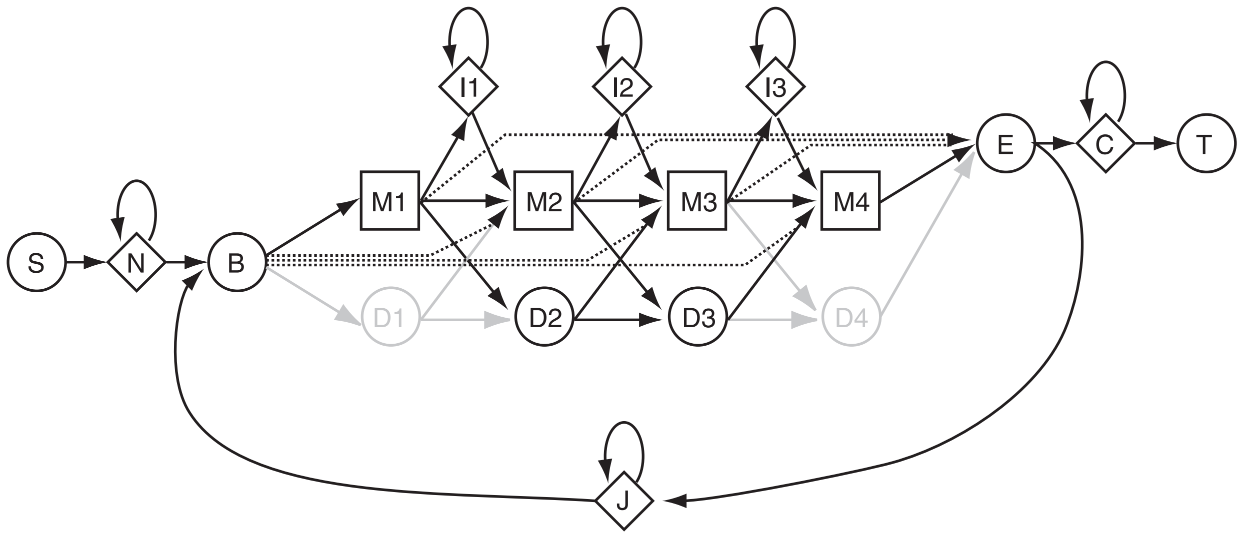



Figure 2.1: *The Plan7 architecture. Squares indicate match states (modeling consensus positions in the alignment). Diamonds indicate insert states (modeling insertions relative to consensus) and special random sequence emitting states. Circles indicate delete states (modeling deletions relative to consensus) and special begin/end states. Arrows indicate state transitions. See text for more details.*

The section of the model composed of M, D, and I states, and the B and E states, is essentially a Krogh/Haussler profile HMM. I refer to this as the "main model". A group of three states M/D/I at the same consensus position in the alignment is called a "node". The main model controls the *data dependent* features of the model. The probability parameters in the main model are generally learned from data in a multiple sequence/structure alignment.

Unlike the original Krogh/Haussler and HMMER model architecture, Plan 7 has no D $\rightarrow$ I or I $\rightarrow$ D transitions. This reduction from 9 to 7 transitions per node in the main model is the origin of the codename Plan 7. (The original HMMER architecture is called Plan 9 in parts of the code.)

The other states (S,N,C,T,J) are called "special states". They (combined with special entry probabilities from B and exit probabilities to E) control the *algorithm dependent* features of the model: how likely the model is to generate various sorts of local or multihit alignments. The algorithm dependent parameters are typically not learned from data, but rather set externally by choosing a desired alignment style.

## Local alignments in Plan 7

The Plan 7 architecture models a complete sequence, regardless of how much of that sequence matches the main model. *All alignments to a Plan 7 model are "global" alignments*, but some of the sequence may be assigned to Plan 7 states (N,C,J) that generate "random" sequence that is not aligned to the main model. Thus, the algorithm dependent parts of the model control the *apparent* locality of the alignments.

Local alignments with respect to the sequence (i.e., allowing a match to the main model anywhere internal to a longer sequence) are controlled by the N and C states. If the N $\rightarrow$ N transition is set to 0, alignments are constrained to start in the main model at the very first residue. Similarly, if the C $\rightarrow$ C transition is set to 0, alignments are constrained to match the main model at the very last residue in the sequence.

Local alignments with respect to the model (i.e., allowing fragments of the model to match the sequence) are controlled by B $\rightarrow$ M "entry" transitions and M $\rightarrow$ E "exit" transitions, shown as dotted lines in the Plan

7 figure. Setting all entries but the B $\rightarrow$ $M_1$ transition to 0 forces a partially "global" alignment in which all alignments to the main model must start at the first match or delete state. Setting all exits to 0 but the final M $\rightarrow$ E transition (which is always 1.0) forces a partially global alignment in which all alignments to the main model must end at the final match or delete state.

Multiple hit alignments are controlled by the E $\rightarrow$ J transition and the J state. If the E $\rightarrow$ J transition is set to 0, a sequence may only contain one domain (one alignment to the main model). If it is nonzero, more than one domain per sequence can be aligned to the main model. The J $\rightarrow$ J transition controls the expected length of the intervening sequence between domains; the lower this probability, the more clustered the domains are expected to be.

The original HMMER1 search programs are encoded in Plan 7 models as follows:

**hmms** Simple global alignment. $t_{NN}$ and $t_{CC}$ set to $t_{GG}$ from the null model (see below). $t_{EJ}$ set to zero. Internal entries and exits set to zero.

**hmmls** Akin to "profile" alignment (as Waterman and others call it): global with respect to main model, local with respect to sequence, multiple nonoverlapping domains allowed. $t_{NN}$ and $t_{CC}$ set to $t_{GG}$ from the null model. $t_{EJ}$ set to 0.5. Internal entries and exits set to zero.

**hmmsw** Smith/Waterman alignment: fully local with respect to both the main model and the sequence; single domain. $t_{NN}$ and $t_{CC}$ set to $t_{GG}$ from the null model. $t_{EJ}$ set to zero. Internal entries are set to $0.5/(M - 1)$ for a model with $M$ match states. Exit probabilities are set such that the overall chance of exiting from an internal match state is 0.5, and the posterior probability distribution over which match state is exited is equal (I'm trying to word this carefully; the M $\rightarrow$ E probabilities are not equal, because there's a small compensation for the fact that if you can leave from $M_5$, then the chance that you'll even get to $M_6$ is reduced.)

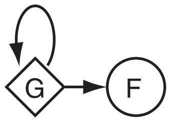
**hmmfs** Multihit Smith/Waterman alignment. Same as the above, but $t_{EJ}$ is set to 0.5.

One advantage of Plan 7 is great flexibility in choosing an alignment style. Complicated alignment styles are easily encoded in the model parameters without changing the alignment algorithm. For example, say you wanted to model human L1 retrotransposon elements. Because of the way L1 elements are inserted by reverse transcriptase (RVT), L1 elements tend to have a defined 3' end (RVT starts replication at the same place in each new L1) but a ragged 5' end (RVT prematurely falls off a new L1 in an unpredictable fashion). A specialized L1 model could define non-zero internal entry probabilities and zero internal exit probabilities to model this biological situation.

One disadvantage of Plan 7 is that if you decide you want to do both local and global alignments, you need two different models (or you need to do one search, then change the model). This wouldn't be a terrible burden except for the fact that the algorithm-dependent parameters strongly affect the values of the $\mu$ and $\lambda$ parameters that E-value statistics depend on. If the algorithm dependent parameters are changed, these parameters are lost and the model should be recalibrated with `hmmcalibrate` – and `hmmcalibrate` is relatively slow.

## The Plan 7 null model

When HMM alignments are scored, they are scored by a log-odds score relative to a "null model" of random sequence composition (Barrett et al., 1997). In Plan 7, this model is now specified as a full probabilistic model too:




The G state has a symbol emission probability distribution for $K$ symbols in the alphabet. By default, this distribution is set either to the average amino acid composition of SWISSPROT 34, or to 0.25 for each nucleotide. The G $\rightarrow$ G transition controls the expected length of observed random sequences; in practice, this transition probability is so close to 1 that it has very little effect. The F state is just a dummy end state like the T state in the Plan 7 architecture.

### Wing retraction in Plan 7 dynamic programming

In the figure of the Plan 7 architecture, you may have noticed that the first and last delete states are greyed out. Internally in HMMER, these delete states exist in the "probability form" of the model (when the model is being worked with in every way except alignments) but they are carefully removed in the "search form" of the model (when the model is converted to log-odds scores and used for alignments). This process is called "wing retraction" in the code, by analogy to a swept-wing fighter changing from a wings-out takeoff and landing configuration to a wings-back configuration for high speed flight.

The problem is that the Plan 7 model allows cycles through the J state. If a continuous nonemitting "mute cycle" were possible (J, B, D states, E, and back to J), dynamic programming recursions would fail. This is why special mute states like delete states must be handled carefully in HMM dynamic programming algorithms; see (Durbin et al., 1998) for further discussion. The easiest way to prevent a mute cycle is to make sure that the model must pass through at least one match state per path through the main model.

Wing retraction involves folding the probabilities of the terminal delete paths into the Plan 7 entry and exit probabilities. For example, in wing retraction the "algorithm dependent" B $\rightarrow$ $M_3$ entry probability is incremented by the probability of the "data dependent" path B $\rightarrow D_1 \rightarrow D_2 \rightarrow M_3$.

Having the wing retraction step, rather than *always* folding these probabilities together, is a design decision, preserving a distinction between the "algorithm dependent" and "data dependent" parts of the model.

## 2.4 Sequence file formats

For all the programs, unaligned sequence files can be in FASTA, Genbank, EMBL, or SWISS-PROT format, as well as a few other common file formats. The programs automatically detect what format the file is in and whether the sequences are DNA, RNA, or protein.

Aligned sequence files can be in ClustalW, GCG MSF, or SELEX format. SELEX format is a simple format of one line per sequence, containing the name first, followed by the aligned sequence. ClustalW, MSF and SELEX alignment files can also be used where unaligned format files are required; the sequences will be read in and their gaps removed.

Full specifications of these file formats and the other formats recognized by the HMM package are in the file formats chapter near the end of the guide.

The programs work on RNA, DNA, and protein sequence. They automatically detect what your sequences are. The behavior of the programs when a nucleic acid model is used to analyze protein sequences, or vice versa, is undefined. Certain other situations may arise (trying to search the "complementary strand" of a protein database, for example) that are nonsensical in certain contexts. Be forewarned. If you're lucky, the software will issue a snide warning to you if you try to do something nonsensical, but usually it will assume you know what you're doing.

## 2.5 Command line options

If you forget the command-line syntax or available options of any of the programs, you can type the name of the program with no other arguments and get a short help message, including summaries of the common options.

Commonly used options are generally small letters, like `-a`. More infrequently used options are generally large letters, like `-A`. Expert or experimental options are generally in the GNU long form, like `--null2`.

If you call any program with an option `-h`, you get an augmented help message, including version info (the software version number is helpful if you report bugs or other problems to me) and a complete summary of all the available options, including rarely used and expert/experimental ones.

## 2.6 Environment variables

HMMER is built to coexist peacefully with the BLAST suite of database search programs (Altschul, 1991). HMMER reads the following environment variables (the examples given use UNIX csh syntax):

**BLASTDB** Location of sequence databases that `hmmsearch` will look in, in addition to the current working directory. Multiple directories are allowed, separated by colons. A trailing slash on each path is important to BLAST, but not to HMMER.
Examples:
> **setenv BLASTDB /nfs/databases/**
> **setenv BLASTDB /nfs/databases/:/nfs/moredatabases/**

**BLASTMAT** Location of substitution matrices that `hmmbuild --pam` (the PAM prior option) can read. Although HMMER can parse a colon-separated list, BLAST must have a single directory path here. Example:
> **setenv BLASTMAT /nfs/databases/matrix/**

**HMMERDB** Location of HMMs, PFAM, or other HMMER specific data files. Any program that reads an HMM file looks in both HMMERDB and the current working directory. Multiple directories are allowed, colon-separated. Examples:
> **setenv HMMERDB /usr/local/lib/hmmer/**
> **setenv HMMERDB /usr/local/lib/hmmer/:/nfs/databases/pfam/**

**HMMER_NCPU** On multiprocessors that support POSIX threads (this includes almost all modern UNIX multiprocessors; for example, SGI Origin servers), the programs `hmmcalibrate`, `hmmpfam`, and `hmmsearch` run as parallelized, multithreaded applications. Normally they will take over all available CPUs in the machine. HMMER_NCPU sets a maximum number of CPUs to utilize, so HMMER searches are "good citizens", leaving some CPU power for other jobs. An example of configuring HMMER to use only 16 processors on a 32-processor Origin:
> **setenv HMMER_NCPU 16**

## 2.7 Other profile HMM implementations

Several implementations of profile HMM methods and related PSSM methods are available. Some are listed in the table below.

| Software | URL |
|---|---|
| HMMER | http://hmmer.wustl.edu/ |
| SAM | http://www.cse.ucsc.edu/research/compbio/sam.html |
| PFTOOLS | http://ulrec3.unil.ch:80/profile/ |
| HMMpro | http://www.netid.com/html/hmmpro.html |
| GENEWISE | http://www.sanger.ac.uk/Software/Wise2/ |
| PROBE | ftp://ncbi.nlm.nih.gov/pub/neuwald/probe1.0/ |
| META-MEME | http://www.cse.ucsd.edu/users/bgrundy/metameme.1.0.html |
| BLOCKS | http://www.blocks.fhcrc.org/ |
| PSI-BLAST | http://www.ncbi.nlm.nih.gov/BLAST/newblast.html |

HMMER, SAM, PFTOOLS, and HMMpro are the most closely related to the profile HMM methods introduced by Krogh et al. HMMpro is commercial, not free software.

# Chapter 3

# Manual pages

## 3.1 HMMER - profile hidden Markov model software

### Synopsis

**hmmalign** Align multiple sequences to a profile HMM.

**hmmbuild** Build a profile HMM from a given multiple sequence alignment.

**hmmcalibrate** Determine appropriate statistical significance parameters for a profile HMM prior to doing database searches.

**hmmconvert** Convert HMMER profile HMMs to other formats, such as GCG profiles.

**hmmemit** Generate sequences probabilistically from a profile HMM.

**hmmpfam** Search a profile HMM database with a sequence (i.e., annotate various kinds of domains in the query sequence).

**hmmsearch** Search a sequence database with a profile HMM (i.e., find additional homologues of a modeled family).

### Description

These programs use profile hidden Markov models (profile HMMs) to model the primary structure consensus of a family of protein or nucleic acid sequences.

## Options

All **HMMER** programs give a brief summary of their command-line syntax and options if invoked without any arguments. When invoked with the single argument, **-h** (i.e., help), a program will report more verbose command-line usage information, including rarely used, experimental, and expert options. **-h** will report version numbers which are useful if you need to report a bug or problem to me.

Each **HMMER** program has its own man page briefly summarizing command line usage. There is also a user's guide that came with the software distribution, which includes a tutorial introduction and more detailed descriptions of the programs.

See http://hmmer.wustl.edu/ for on-line documentation and the current HMMER release.

In general, no command line options should be needed by beginning users. The defaults are set up for optimum performance in most situations. Options that are single lowercase letters (e.g. **-a** ) are "common" options that are expected to be frequently used and will be important in many applications. Options that are single uppercase letters (e.g. **-B** ) are usually less common options, but also may be important in some applications. Options that are full words (e.g. **--verbose** ) are either rarely used, experimental, or expert options. Some experimental options are only there for my own ongoing experiments with HMMER, and may not be supported or documented adequately.

## Sequence File Formats

In general, **HMMER** attempts to read most common biological sequence file formats. It autodetects the format of the file. It also autodetects whether the sequences are protein or nucleic acid. Standard IUPAC degeneracy codes are allowed in addition to the usual 4-letter or 20-letter codes.

**Unaligned sequences** Unaligned sequence files may be in FASTA, Swissprot, EMBL, GenBank, PIR, Intelligenetics, Strider, or GCG format. These formats are documented in the User's Guide.

**Sequence alignments** Multiple sequence alignments may be in CLUSTALW, SELEX, or GCG MSF format. These formats are documented in the User's Guide.

## Environment Variables

For ease of using large stable sequence and HMM databases, **HMMER** looks for sequence files and HMM files in the current working directory as well as in system directories specified by environment variables.

**BLASTDB** Specifies the directory location of sequence databases. Example: **/seqlibs/blastdb/.** In installations that use BLAST software, this environment variable is likely to already be set.

**HMMERDB** Specifies the directory location of HMM databases. Example: **/seqlibs/pfam/.**

## 3.2 `hmmalign` - align sequences to an HMM profile

### Synopsis

**hmmalign** *[options] hmmfile seqfile*

### Description

**hmmalign** reads an HMM file from *hmmfile* and a set of sequences from *seqfile,* aligns the sequences to the profile HMM, and outputs a multiple sequence alignment.

*seqfile* may be in any unaligned or aligned file format accepted by HMMER. If it is in a multiple alignment format (e.g. MSF, SELEX, ClustalW), the existing alignment is ignored.

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-m** Include in the alignment only those symbols aligned to match states. Do not show symbols assigned to insert states.

**-o** <*f*> Save alignment to file <*f*> instead of to standard output.

**-q** quiet; suppress all output except the alignment itself. Useful for piping or redirecting the output.

### Expert Options

**--mapali** <*f*> Reads an alignment from file <*f*> and aligns it as a single object to the HMM; e.g. the alignment in <*f*> is held fixed. This allows you to align sequences to a model with **hmmalign** and view them in the context of an existing trusted multiple alignment. The alignment to the alignment is defined by a "map" kept in the HMM, and so is fast and guaranteed to be consistent with the way the HMM was constructed from the alignment. The alignment in the file <*f*> must be exactly the alignment that the HMM was built from. Compare the **--withali** option.

**--withali** <*f*> Reads an alignment from file <*f*> and aligns it as a single object to the HMM; e.g. the alignment in <*f*> is held fixed. This allows you to align sequences to a model with **hmmalign** and view them in the context of an existing trusted multiple alignment. The alignment to the alignment is done with a heuristic (nonoptimal) dynamic programming procedure, which may be somewhat slow and is not guaranteed to be completely consistent with the way the HMM was constructed (though it should be quite close). However, any alignment can be used, not just the alignment that the HMM was built from. Compare the **--mapali** option.

## 3.3 `hmmbuild` - build a profile HMM from an alignment

### Synopsis

**hmmbuild** *[options] hmmfile alignfile*

### Description

**hmmbuild** reads a multiple sequence alignment file *alignfile* , builds a new profile HMM, and saves the HMM in *hmmfile.*

*alignfile* may be in ClustalW, GCG MSF, or SELEX alignment format.

By default, the model is configured to find one or more nonoverlapping alignments to the complete model. This is analogous to the behavior of the **hmmls** program of HMMER 1. To configure the model for a single global alignment, use the **-g** option; to configure the model for multiple local alignments a la the old program **hmmfs,** use the **-f** option; and to configure the model for a single local alignment (a la standard Smith/Waterman, or the old **hmmsw** program), use the **-s** option.

### Options

**-f** Configure the model for finding multiple domains per sequence, where each domain can be a local (fragmentary) alignment. This is analogous to the old **hmmfs** program of HMMER 1.

**-g** Configure the model for finding a single global alignment to a target sequence, analogous to the standard Needleman/Wunsch algorithm or the old **hmms** program of HMMER 1.

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-n** *<s>* Name this HMM *<s>*. *<s>* can be any string of non-whitespace characters (e.g. one "word"). There is no length limit (at least not one imposed by HMMER; your shell will complain about command line lengths first).

**-o** *<f>* Re-save the starting alignment to *<f>*, in SELEX format. The columns which were assigned to match states will be marked with x's in an #=RF annotation line. If either the **--hand** or **--fast** construction options were chosen, the alignment may have been slightly altered to be compatible with Plan 7 transitions, so saving the final alignment and comparing to the starting alignment can let you view these alterations. See the User's Guide for more information on this arcane side effect.

**-s** Configure the model for finding a single local alignment per target sequence. This is analogous to the standard Smith/Waterman algorithm or the **hmmsw** program of HMMER 1.

**-A** Append this model to an existing *hmmfile* rather than creating *hmmfile.* Useful for building HMM libraries (like Pfam).

**-F** Force overwriting of an existing *hmmfile.* Otherwise HMMER will refuse to clobber your existing HMM files, for safety's sake.

## Expert Options

**--amino** Force the sequence alignment to be interpreted as amino acid sequences. Normally HMMER autodetects whether the alignment is protein or DNA, but sometimes alignments are so small that autodetection is ambiguous. See **--nucleic.**

**--archpri** <*x*> Set the "architecture prior" used by MAP architecture construction to <*x*>, where <*x*> is a probability between 0 and 1. This parameter governs a geometric prior distribution over model lengths. As <*x*> increases, longer models are favored a priori. As <*x*> decreases, it takes more residue conservation in a column to make a column a "consensus" match column in the model architecture. The 0.85 default has been chosen empirically as a reasonable setting.

**--binary** Write the HMM to *hmmfile* in HMMER binary format instead of readable ASCII text.

**--cfile** <*f*> Save the observed emission and transition counts to <*f*> after the architecture has been determined (e.g. after residues/gaps have been assigned to match, delete, and insert states). This option is used in HMMER development for generating data files useful for training new Dirichlet priors. The format of count files is documented in the User's Guide.

**--fast** Quickly and heuristically determine the architecture of the model by assigning all columns will more than a certain fraction of gap characters to insert states. By default this fraction is 0.5, and it can be changed using the **--gapmax** option. The default construction algorithm is a maximum a posteriori (MAP) algorithm, which is slower.

**--gapmax** <*x*> Controls the *--fast* model construction algorithm, but if *--fast* is not being used, has no effect. If a column has more than a fraction <*x*> of gap symbols in it, it gets assigned to an insert column. <*x*> is a frequency from 0 to 1, and by default is set to 0.5. Higher values of <*x*> mean more columns get assigned to consensus, and models get longer; smaller values of <*x*> mean fewer columns get assigned to consensus, and models get smaller. <*x*>

**--hand** Specify the architecture of the model by hand: the alignment file must be in SELEX format, and the #=RF annotation line is used to specify the architecture. Any column marked with a non-gap symbol (such as an 'x', for instance) is assigned as a consensus (match) column in the model.

**--idlevel** <*x*> Controls both the determination of effective sequence number and the behavior of the --*wblosum* weighting option. The sequence alignment is clustered by percent identity, and the number of clusters at a cutoff threshold of <*x*> is used to determine the effective sequence number. Higher values of <*x*> give more clusters and higher effective sequence numbers; lower values of <*x*> give fewer clusters and lower effective sequence numbers. <*x*> is a fraction from 0 to 1, and by default is set to 0.62 (corresponding to the clustering level used in constructing the BLOSUM62 substitution matrix).

**--noeff** Turn off the effective sequence number calculation, and use the true number of sequences instead. This will usually reduce the sensitivity of the final model (so don't do it without good reason!)

**--nucleic** Force the alignment to be interpreted as nucleic acid sequence, either RNA or DNA. Normally HMMER autodetects whether the alignment is protein or DNA, but sometimes alignments are so small that autodetection is ambiguous. See **--amino.**

**--null** <*f*> Read a null model from <*f*>. The default for protein is to use average amino acid frequencies from Swissprot 34 and p1 = 350/351; for nucleic acid, the default is to use 0.25 for each base and p1 = 1000/1001. For documentation of the format of the null model file and further explanation of how the null model is used, see the User's Guide.

**--pam** <*f*> Apply a heuristic PAM- (substitution matrix-) based prior instead of the default mixture Dirichlet. The substitution matrix is read from <*f*>. See **--pamwgt.**

**--pamwgt** <*x*> Controls the weight on a PAM-based prior. Only has effect if **--pam** option is also in use. <*x*> is a positive real number, 20.0 by default. <*x*> is the number of "pseudocounts" contriubuted by the heuristic prior. Very high values of <*x*> can force a scoring system that is entirely driven by the substitution matrix, making HMMER somewhat approximate Gribskov profiles.

**--prior** <*f*> Read a Dirichlet prior from <*f*>, replacing the default mixture Dirichlet. The format of prior files is documented in the User's Guide, and an example is given in the Demos directory of the HMMER distribution.

**--swentry** <*x*> Controls the total probability that is distributed to local entries into the model, versus starting at the beginning of the model as in a global alignment. <*x*> is a probability from 0 to 1, and by default is set to 0.5. Higher values of <*x*> mean that hits that are fragments on their left (N or 5'-terminal) side will be penalized less, but complete global alignments will be penalized more. Lower values of <*x*> mean that fragments on the left will be penalized more, and global alignments on this side will be favored. This option only affects the configurations that allow local alignments, e.g. **-s** and **-f;** unless one of these options is also activated, this option has no effect. You have independent control over local/global alignment behavior for the N/C (5'/3') termini of your target sequences using **--swentry** and **--swexit.**

**--swexit** $<x>$ Controls the total probability that is distributed to local exits from the model, versus ending an alignment at the end of the model as in a global alignment. $<x>$ is a probability from 0 to 1, and by default is set to 0.5. Higher values of $<x>$ mean that hits that are fragments on their right (C or 3'-terminal) side will be penalized less, but complete global alignments will be penalized more. Lower values of $<x>$ mean that fragments on the right will be penalized more, and global alignments on this side will be favored. This option only affects the configurations that allow local alignments, e.g. **-s** and **-f;** unless one of these options is also activated, this option has no effect. You have independent control over local/global alignment behavior for the N/C (5'/3') termini of your target sequences using **--swentry** and **--swexit.**

**--verbose** Print more possibly useful stuff, such as the individual scores for each sequence in the alignment.

**--wblosum** Use the BLOSUM filtering algorithm to weight the sequences, instead of the default. Cluster the sequences at a given percentage identity (see **--idlevel);** assign each cluster a total weight of 1.0, distributed equally amongst the members of that cluster.

**--wgsc** Use the Gerstein/Sonnhammer/Chothia ad hoc sequence weighting algorithm. This is already the default, so this option has no effect (unless it follows another option in the --w family, in which case it overrides it).

**--wme** Use the Krogh/Mitchison maximum entropy algorithm to "weight" the sequences. This supercedes the Eddy/Mitchison/Durbin maximum discrimination algorithm, which gives almost identical weights but is less robust. ME weighting seems to give a marginal increase in sensitivity over the default GSC weights, but takes a fair amount of time.

**--wnone** Turn off all sequence weighting.

**--wvoronoi** Use the Sibbald/Argos Voronoi sequence weighting algorithm in place of the default GSC weighting.

## 3.4 `hmmcalibrate` - calibrate HMM search statistics

### Synopsis

**hmmcalibrate** *[options] hmmfile*

### Description

**hmmcalibrate** reads an HMM file from *hmmfile,* scores a large number of synthesized random sequences with it, fits an extreme value distribution (EVD) to the histogram of those scores, and re-saves *hmmfile* now including the EVD parameters.

**hmmcalibrate** may take several minutes (or longer) to run. While it is running, a temporary file called *hmmfile.xxx* is generated in your working directory. If you abort **hmmcalibrate** prematurely (ctrl-C, for instance), your original *hmmfile* will be untouched, and you should delete the *hmmfile.xxx* temporary file.

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

### Expert Options

**--cpu** <*n*> Sets the maximum number of CPUs that the program will run on. The default is to use all CPUs in the machine. Overrides the HMMER_NCPU environment variable. Only affects threaded versions of HMMER (the default on most systems).

**--fixed** <*n*> Fix the length of the random sequences to <*n*>, where <*n*> is a positive (and reasonably sized) integer. The default is instead to generate sequences with a variety of different lengths, controlled by a Gaussian (normal) distribution.

**--histfile** <*f*> Save a histogram of the scores and the fitted theoretical curve to file <*f*>.

**--mean** <*x*> Set the mean length of the synthetic sequences to <*x*>, where <*x*> is a positive real number. The default is 350.

**--num** <*n*> Set the number of synthetic sequences to <*n*>, where <*n*> is a positive integer. If <*n*> is less than about 1000, the fit to the EVD may fail. Higher numbers of <*n*> will give better determined EVD parameters. The default is 5000; it was empirically chosen as a tradeoff between accuracy and computation time.

**--pvm** Run on a Parallel Virtual Machine (PVM). The PVM must already be running. The client program **hmmcalibrate-pvm** must be installed on all the PVM nodes. Optional PVM support must have been compiled into HMMER.

**--sd** $<x>$ Set the standard deviation of the synthetic sequence length distribution to $<x>$, where $<x>$ is a positive real number. The default is 350. Note that the Gaussian is left-truncated so that no sequences have lengths $<= 0$.

**--seed** $<n>$ Set the random seed to $<n>$, where $<n>$ is a positive integer. The default is to use **time()** to generate a different seed for each run, which means that two different runs of **hmmcalibrate** on the same HMM will give slightly different results. You can use this option to generate reproducible results for different **hmmcalibrate** runs on the same HMM.

## 3.5 `hmmconvert` - convert between profile HMM file formats

### Synopsis

**hmmconvert** *[options] oldhmmfile newhmmfile*

### Description

**hmmconvert** reads an HMM file from *oldhmmfile* in any HMMER format, and writes it to a new file *newhmmfile* in a new format. *oldhmmfile* and *newhmmfile* must be different files; you can't reliably overwrite the old file. By default, the new HMM file is written in HMMER 2 ASCII format.

Available formats are HMMER 2 ASCII (default), HMMER 2 binary *(-b)* GCG profile *(-p)* , and Compugen XSW extended profile *(-P).*

### Options

**-a** Convert to HMMER 2 ASCII file. This is the default, so this option is unnecessary.

**-b** Convert to HMMER 2 binary file.

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-p** Convert to GCG profile .prf format.

**-A** Append mode; append to *newhmmfile* rather than creating a new file.

**-F** Force; if *newhmmfile* already exists, and *-A* is not being used to append to the file, hmmconvert will refuse to clobber the existing file unless *-F* is used.

**-P** Convert the HMM to Compugen XSW extended profile format, which is similar to GCG profile format but has two extra columns for delete-open and delete-extend costs. (I do not believe that Compugen publicly supports this format; it may be undocumented.)

## 3.6 `hmmemit` - generate sequences from a profile HMM

### Synopsis

**hmmemit** *[options] hmmfile*

### Description

**hmmemit** reads an HMM file from *hmmfile* and generates a number of sequences from it; or, if the **-c** option is selected, generate a single majority-rule consensus. This can be useful for various applications in which one needs a simulation of sequences consistent with a sequence family consensus.

By default, **hmmemit** generates 10 sequences and outputs them in FASTA (unaligned) format.

### Options

**-a** Write the generated sequences in an aligned format (SELEX) rather than FASTA.

**-c** Predict a single majority-rule consensus sequence instead of sampling sequences from the HMM's probability distribution. Highly conserved residues (p >= 0.9 for DNA, p >= 0.5 for protein) are shown in upper case; others are shown in lower case. Some insert states may become part of the majority rule consensus, because they are used in >= 50% of generated sequences; when this happens, insert-generated residues are simply shown as ”x”.

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-n** *<n>* Generate *<n>* sequences. Default is 10.

**-o** *<f>* Save the sequences to file *<f>* rather than writing them to stdout.

**-q** Quiet; suppress all output except for the sequences themselves. Useful for piping or directing the output.

### Expert Options

**--seed** *<n>* Set the random seed to *<n>*, where *<n>* is a positive integer. The default is to use **time()** to generate a different seed for each run, which means that two different runs of **hmmemit** on the same HMM will give slightly different results. You can use this option to generate reproducible results.

## 3.7 `hmmfetch` - retrieve an HMM from an HMM database

### Synopsis

**hmmfetch** *[options] database name*

### Description

**hmmfetch** is a small utility that retrieves an HMM called *name* from a HMMER model database called *database.* in a new format, and prints that model to standard output. For example, *hmmfetch* Pfam rrm retrieves the RRM (RNA recognition motif) model from Pfam, if the environment variable HMMERDB is set to the location of the Pfam database. The retrieved HMM file is written in HMMER 2 ASCII format.

The database must have an associated GSI index file. To index an HMM database, use the program **hmmindex.**

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

## 3.8 `hmmindex` - create a binary GSI index for an HMM database

### Synopsis

**hmmindex** *[options] database*

### Description

**hmmindex** is a utility that creates a binary GSI ("generic sequence index" format) index for an HMM database file called *database*. The new index file is named *database.*gsi. A GSI index file is required for **hmmfetch** to work, and also for the PVM implementation of **hmmpfam.**

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

## 3.9 `hmmpfam` - search a single sequence against an HMM database

### Synopsis

**hmmpfam** *[options] hmmfile seqfile*

### Description

**hmmpfam** reads a single sequence from *seqfile* and compares it against all the HMMs in *hmmfile* looking for significantly similar sequence matches.

*hmmfile* will be looked for first in the current working directory, then in a directory named by the environment variable *HMMERDB*. This lets administrators install HMM library(s) such as Pfam in a common location.

The output consists of three sections: a ranked list of the best scoring HMMs, a list of the best scoring domains in order of their occurrence in the sequence, and alignments for all the best scoring domains. A sequence score may be higher than a domain score for the same sequence if there is more than one domain in the sequence; the sequence score takes into account all the domains. All sequences scoring above the *-E* and *-T* cutoffs are shown in the first list, then *every* domain found in this list is shown in the second list of domain hits. If desired, E-value and bit score thresholds may also be applied to the domain list using the *--domE* and *--domT* options.

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-n** Specify that models and sequence are nucleic acid, not protein. Other HMMER programs autodetect this; but because of the order in which **hmmpfam** accesses data, it can't reliably determine the correct "alphabet" by itself.

**-A** <*n*> Limits the alignment output to the <*n*> best scoring domains. **-A0** shuts off the alignment output and can be used to reduce the size of output files.

**-E** <*x*> Set the E-value cutoff for the per-sequence ranked hit list to <*x*>, where <*x*> is a positive real number. The default is 10.0. Hits with E-values better than (less than) this threshold will be shown.

**-T** <*x*> Set the bit score cutoff for the per-sequence ranked hit list to <*x*>, where <*x*> is a real number. The default is negative infinity; by default, the threshold is controlled by E-value and not by bit score. Hits with bit scores better than (greater than) this threshold will be shown.

**-Z** <*n*> Calculate the E-value scores as if we had seen a sequence database of <*n*> sequences. The default is arbitrarily set to 59021, the size of Swissprot 34.

## Expert Options

**--cpu** <*n*> Sets the maximum number of CPUs that the program will run on. The default is to use all CPUs in the machine. Overrides the HMMER_NCPU environment variable. Only affects threaded versions of HMMER (the default on most systems).

**--domE** <*x*> Set the E-value cutoff for the per-domain ranked hit list to <*x*>, where <*x*> is a positive real number. The default is infinity; by default, all domains in the sequences that passed the first threshold will be reported in the second list, so that the number of domains reported in the per-sequence list is consistent with the number that appear in the per-domain list.

**--domT** <*x*> Set the bit score cutoff for the per-domain ranked hit list to <*x*>, where <*x*> is a real number. The default is negative infinity; by default, all domains in the sequences that passed the first threshold will be reported in the second list, so that the number of domains reported in the per-sequence list is consistent with the number that appear in the per-domain list. *Important* note: only one domain in a sequence is absolutely controlled by this parameter, or by **--domT.** The second and subsequent domains in a sequence have a de facto bit score threshold of 0 because of the details of how HMMER works. HMMER requires at least one pass through the main model per sequence; to do more than one pass (more than one domain) the multidomain alignment must have a better score than the single domain alignment, and hence the extra domains must contribute positive score. See the Users' Guide for more detail.

**--forward** Use the Forward algorithm instead of the Viterbi algorithm to determine the per-sequence scores. Per-domain scores are still determined by the Viterbi algorithm. Some have argued that Forward is a more sensitive algorithm for detecting remote sequence homologues; my experiments with HMMER have not confirmed this, however.

**--null2** Turn off the post hoc second null model. By default, each alignment is rescored by a postprocessing step that takes into account possible biased composition in either the HMM or the target sequence. This is almost essential in database searches, especially with local alignment models. There is a very small chance that this postprocessing might remove real matches, and in these cases **--null2** may improve sensitivity at the expense of reducing specificity by letting biased composition hits through.

**--pvm** Run on a Parallel Virtual Machine (PVM). The PVM must already be running. The client program **hmmpfam-pvm** must be installed on all the PVM nodes. The HMM database *hmmfile* and an associated GSI index file *hmmfile*.gsi must also be installed on all the PVM nodes. (The GSI index is produced by the program **hmmindex**.) Because the PVM implementation is I/O bound, it is highly recommended that each node have a local copy of *hmmfile* rather than NFS mounting a shared copy. Optional PVM support must have been compiled into HMMER for **--pvm** to function.

**--xnu** Turn on XNU filtering of target protein sequences. Has no effect on nucleic acid sequences. In trial experiments, **--xnu** appears to perform less well than the default post hoc null2 model.

## 3.10 `hmmsearch` - search a sequence database with a profile HMM

### Synopsis

**hmmsearch** *[options] hmmfile seqfile*

### Description

**hmmsearch** reads an HMM from *hmmfile* and searches *seqfile* for significantly similar sequence matches.

*seqfile* will be looked for first in the current working directory, then in a directory named by the environment variable *BLASTDB*. This lets users use existing BLAST databases, if BLAST has been configured for the site.

**hmmsearch** may take minutes or even hours to run, depending on the size of the sequence database. It is a good idea to redirect the output to a file.

The output consists of four sections: a ranked list of the best scoring sequences, a ranked list of the best scoring domains, alignments for all the best scoring domains, and a histogram of the scores. A sequence score may be higher than a domain score for the same sequence if there is more than one domain in the sequence; the sequence score takes into account all the domains. All sequences scoring above the *-E* and *-T* cutoffs are shown in the first list, then *every* domain found in this list is shown in the second list of domain hits. If desired, E-value and bit score thresholds may also be applied to the domain list using the *--domE* and *--domT* options.

### Options

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-A** <*n*> Limits the alignment output to the <*n*> best scoring domains. **-A0** shuts off the alignment output and can be used to reduce the size of output files.

**-E** <*x*> Set the E-value cutoff for the per-sequence ranked hit list to <*x*>, where <*x*> is a positive real number. The default is 10.0. Hits with E-values better than (less than) this threshold will be shown.

**-T** <*x*> Set the bit score cutoff for the per-sequence ranked hit list to <*x*>, where <*x*> is a real number. The default is negative infinity; by default, the threshold is controlled by E-value and not by bit score. Hits with bit scores better than (greater than) this threshold will be shown.

**-Z** <*n*> Calculate the E-value scores as if we had seen a sequence database of <*n*> sequences. The default is the number of sequences seen in your database file <*seqfile*>.

### Expert Options

**--cpu** <*n*> Sets the maximum number of CPUs that the program will run on. The default is to use all CPUs in the machine. Overrides the HMMER_NCPU environment variable. Only affects threaded versions of HMMER (the default on most systems).

**--domE** <*x*> Set the E-value cutoff for the per-domain ranked hit list to <*x*>, where <*x*> is a positive real number. The default is infinity; by default, all domains in the sequences that passed the first threshold will be reported in the second list, so that the number of domains reported in the per-sequence list is consistent with the number that appear in the per-domain list.

**--domT** <*x*> Set the bit score cutoff for the per-domain ranked hit list to <*x*>, where <*x*> is a real number. The default is negative infinity; by default, all domains in the sequences that passed the first threshold will be reported in the second list, so that the number of domains reported in the per-sequence list is consistent with the number that appear in the per-domain list. *Important* note: only one domain in a sequence is absolutely controlled by this parameter, or by **--domT.** The second and subsequent domains in a sequence have a de facto bit score threshold of 0 because of the details of how HMMER works. HMMER requires at least one pass through the main model per sequence; to do more than one pass (more than one domain) the multidomain alignment must have a better score than the single domain alignment, and hence the extra domains must contribute positive score. See the Users' Guide for more detail.

**--forward** Use the Forward algorithm instead of the Viterbi algorithm to determine the per-sequence scores. Per-domain scores are still determined by the Viterbi algorithm. Some have argued that Forward is a more sensitive algorithm for detecting remote sequence homologues; my experiments with HMMER have not confirmed this, however.

**--null2** Turn off the post hoc second null model. By default, each alignment is rescored by a postprocessing step that takes into account possible biased composition in either the HMM or the target sequence. This is almost essential in database searches, especially with local alignment models. There is a very small chance that this postprocessing might remove real matches, and in these cases **--null2** may improve sensitivity at the expense of reducing specificity by letting biased composition hits through.

**--pvm** Run on a Parallel Virtual Machine (PVM). The PVM must already be running. The client program **hmmsearch-pvm** must be installed on all the PVM nodes. Optional PVM support must have been compiled into HMMER.

**--xnu** Turn on XNU filtering of target protein sequences. Has no effect on nucleic acid sequences. In trial experiments, **--xnu** appears to perform less well than the default post hoc null2 model.

## 3.11 `alistat` - show statistics for a multiple alignment file

### Synopsis

**alistat** *[options] alignfile*

### Description

**alistat** reads a multiple sequence alignment from the file *alignfile* in any supported format (including SELEX, GCG MSF, and CLUSTAL), and shows a number of simple statistics about it. These statistics include the name of the format, the number of sequences, the total number of residues, the average and range of the sequence lengths, the alignment length (e.g. including gap characters).

Also shown are some percent identities. A percent pairwise alignment identity is defined as *(idents* / MIN(len1, len2)) where *idents* is the number of exact identities and *len1,* len2 are the unaligned lengths of the two sequences. The "average percent identity", "most related pair", and "most unrelated pair" of the alignment are the average, maximum, and minimum of all (N)(N-1)/2 pairs, respectively. The "most distant seq" is calculated by finding the maximum pairwise identity (best relative) for all N sequences, then finding the minimum of these N numbers (hence, the most outlying sequence).

### Options

**-a** Show additional verbose information: a table with one line per sequence showing name, length, and its highest and lowest pairwise identity. These lines are prefixed with a * character to enable easily **grep'**ing them out and sorting them. For example, *alistat* -a foo.slx | grep "*" | sort -n +3 gives a ranked list of the most distant sequences in the alignment. Incompatible with the **-f** option.

**-f** Fast; use a sampling method to estimate the average %id. When this option is chosen, **alistat** doesn't show the other three pairwise identity numbers. This option is useful for very large alignments, for which the full (N)(N-1) calculation of all pairs would be prohibitive (e.g. Pfam's GP120 alignment, with over 10,000 sequences). Incompatible with the **-a** option.

**-h** Print brief help; includes version number and summary of all options, including expert options.

## 3.12 `getseq` - get a sequence from a flatfile database.

### Synopsis

**getseq** *[options] seqname*

### Description

**getseq** retrieves the sequence named *seqname* from a sequence database.

Which database is used is controlled by the **-d** and **-D** options, or "little databases" and "big databases". The directory location of "big databases" can be specified by environment variables, such as $SWDIR for Swissprot, and $GBDIR for Genbank (see **-D** for complete list). A complete file path must be specified for "little databases". By default, if neither option is specified and the name looks like a Swissprot identifier (e.g. it has a _ character), the $SWDIR environment variable is used to attempt to retrieve the sequence *seqname* from Swissprot.

A variety of other options are available which allow retrieval of subsequences (*-f,-t*); retrieval by accession number instead of by name (*-a*); reformatting the extracted sequence into a variety of other formats (*-F*); etc.

If the database has been GSI indexed, sequence retrieval will be extremely efficient; else, retrieval may be painfully slow (the entire database may have to be read into memory to find *seqname*). GSI indexing is recommended for all large or permanent databases.

### Options

**-a** Interpret *seqname* as an accession number, not an identifier.

**-d** *<seqfile>* Retrieve the sequence from a sequence file named *<seqfile>*. If a GSI index *<seqfile>.gsi* exists, it is used to speed up the retrieval.

**-f** *<from>* Extract a subsequence starting from position *<from>,* rather than from 1. See **-t.** If *<from>* is greater than *<to>* (as specified by the **-t** option), then the sequence is extracted as its reverse complement (it is assumed to be nucleic acid sequence).

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-o** *<outfile>* Direct the output to a file named *<outfile>*. By default, output would go to stdout.

**-r** *<newname>* Rename the sequence *<newname>* in the output after extraction. By default, the original sequence identifier would be retained. Useful, for instance, if retrieving a sequence fragment; the coordinates of the fragment might be added to the name (this is what Pfam does).

**-t** <*to*> Extract a subsequence that ends at position <*to*>, rather than at the end of the sequence. See **-f.** If <*to*> is less than <*from*> (as specified by the **-f** option), then the sequence is extracted as its reverse complement (it is assumed to be nucleic acid sequence)

**-D** <*database*> Retrieve the sequence from the main sequence database coded <*database*>. For each code, there is an environment variable that specifies the directory path to that database. Recognized codes and their corresponding environment variables are *-Dsw* (Swissprot, $SWDIR); *-Dpir* (PIR, $PIRDIR); *-Dem* (EMBL, $EMBLDIR); *-Dgb* (Genbank, $GBDIR); *-Dwp* (Wormpep, $WORMDIR); and *-Dowl* (OWL, $OWLDIR). Each database is read in its native flatfile format.

**-F** <*format*> Reformat the extracted sequence into a different format. (By default, the sequence is extracted from the database in the same format as the database.) Available formats are **embl, fasta, genbank, gcg, strider, zuker, ig, pir, squid,** and **raw.**

## 3.13 `seqstat` - show statistics and format for a sequence file

### Synopsis

**seqstat** *[options] seqfile*

### Description

**seqstat** reads a sequence file *seqfile* in any supported format, and shows a number of simple statistics about it. Supported formats include (but are not limited to) the unaligned formats FASTA, Genbank, EMBL, SWISS-PROT, PIR, and GCG, and the aligned formats SELEX, Clustal, and GCG MSF. The printed statistics include the name of the format, the residue type of the first sequence (protein, RNA, or DNA), the number of sequences, the total number of residues, and the average and range of the sequence lengths.

### Options

**-a** Show additional verbose information: a table with one line per sequence showing name, length, and description line. These lines are prefixed with a * character to enable easily **grep'**ing them out and sorting them.

**-h** Print brief help; includes version number and summary of all options, including expert options.

## 3.14 `sreformat` - convert sequence file to different format

### Synopsis

**sreformat** *[options] format seqfile*

### Description

**sreformat** reads the sequence file *seqfile* in any supported format, reformats it into a new format specified by *format,* then prints the reformatted text.

Supported input formats include (but are not limited to) the unaligned formats FASTA, Genbank, EMBL, SWISS-PROT, PIR, and GCG, and the aligned formats SELEX, Clustal, and GCG MSF.

Available unaligned output file format codes include *fasta* (FASTA format); *embl* (EMBL/SWISSPROT format); *genbank* (Genbank format); *gcg* (GCG single sequence format); *gcgdata* (GCG flatfile database format); *strider* (MacStrider format); *zuker* (Zuker MFOLD format); *ig* (Intelligenetics format); *pir* (PIR/CODATA flatfile format); *squid* (an undocumented St. Louis format); *raw* (raw sequence, no other information). The available aligned output file format codes include *selex* (SELEX/HMMER/Pfam annotated alignment format); *msf* (GCG MSF format); and *a2m* (aligned FASTA format, called A2M by the UC Santa Cruz HMM group).

Unaligned format files cannot be reformatted to aligned formats. However, aligned formats can be reformatted to unaligned formats -- gap characters are simply stripped out.

This program was originally named **reformat,** but that name clashes with a GCG program of the same name.

### Options

**-d** DNA; convert U's to T's, to make sure a nucleic acid sequence is shown as DNA not RNA. See **-r.**

**-h** Print brief help; includes version number and summary of all options, including expert options.

**-l** Lowercase; convert all sequence residues to lower case. See **-u.**

**-r** RNA; convert T's to U's, to make sure a nucleic acid sequence is shown as RNA not DNA. See **-d.**

**-u** Uppercase; convert all sequence residues to upper case. See **-l.**

## Expert Options

**--pfam** For SELEX alignment output format only, put the entire alignment in one block (don't wrap into multiple blocks). This is close to the format used internally by Pfam in Stockholm and Cambridge.

**--sam** Try to convert gap characters to UC Santa Cruz SAM style, where a . means a gap in an insert column, and a - means a deletion in a consensus/match column. This only works for converting aligned file formats, and only if the alignment already adheres to the SAM convention of upper case for residues in consensus/match columns, and lower case for residues in insert columns. This is true, for instance, of all alignments produced by old versions of HMMER. (HMMER2 produces alignments that adhere to SAM's conventions even in gap character choice.) This option was added to allow Pfam alignments to be reformatted into something more suitable for profile HMM construction using the UCSC SAM software.

**--samfrac** $<x>$ Try to convert the alignment gap characters and residue cases to UC Santa Cruz SAM style, where a . means a gap in an insert column and a - means a deletion in a consensus/match column, and upper case means match/consensus residues and lower case means inserted resiudes. This will only work for converting aligned file formats, but unlike the **--sam** option, it will work regardless of whether the file adheres to the upper/lower case residue convention. Instead, any column containing more than a fraction $<x>$ of gap characters is interpreted as an insert column, and all other columns are interpreted as match columns. This option was added to allow Pfam alignments to be reformatted into something more suitable for profile HMM construction using the UCSC SAM software.

# Chapter 4

# File formats

## 4.1 HMMER save files

The file `Demos/rrm.hmm` gives an example of a HMMER ASCII save file. An abridged version is shown here, where (...) mark deletions made for clarity and space:

```
HMMER2.0
NAME  rrm
ACC   PF00076
DESC  RNA recognition motif. (aka RRM, RBD, or RNP domain)
LENG  72
ALPH  Amino
RF    no
CS    no
MAP   yes
COM   ../src/hmmbuild -F rrm.hmm rrm.slx
COM   ../src/hmmcalibrate rrm.hmm
NSEQ  70
DATE  Wed Jul  8 08:13:25 1998
CKSUM 2768
GA    13.3 0.0
TC    13.40 0.60
NC    13.20 13.20
XT      -8455     -4  -1000  -1000  -8455     -4  -8455     -4
NULT      -4  -8455
NULE     595  -1558     85    338   -294    453  -1158    (...)
EVD   -49.999123   0.271164
HMM        A      C      D      E      F      G      H      I (...)
         m->m   m->i   m->d   i->m   i->i   d->m   d->d   b->m   m->e
          -21      *  -6129
     1  -1234   -371  -8214  -7849  -5304  -8003  -7706   2384  (...) 1
     -   -149   -500    233     43   -381    399    106   -626  (...)
     -    -11 -11284 -12326   -894  -1115   -701  -1378    -21      *
     2  -3634  -3460  -5973  -5340   3521  -2129  -4036   -831  (...) 2
     -   -149   -500    233     43   -381    399    106   -626  (...)
     -    -11 -11284 -12326   -894  -1115   -701  -1378      *      *
(...)
    71  -1165  -4790   -240   -275  -5105  -4306   1035  -2009  (...) 90
     -   -149   -500    233     43   -381    398    106   -626  (...)
```

```
     -     -43  -6001 -12336   -150  -3342   -701  -1378      *      *
    72  -1929   1218  -1535  -1647  -3990  -4677  -3410   1725  (...) 92
     -      *      *      *      *      *      *      *      *  (...)
     -      *      *      *      *      *      *      *      *      0
//
```

HMMER2 profile HMM save files have a very different format compared to the previous HMMER1 ASCII formats. The HMMER2 format provides all the necessary parameters to compare a protein sequence to a HMM, including the search mode of the HMM (hmmls, hmmfs, hmmsw, and hmms in the old HMMER1 package), the null (background) model, and the statistics to evaluate the match on the basis of a previously fitted extreme value distribution.

The format consists of one or more HMMs. Each HMM starts with the identifier "HMMER2.0" on a line by itself and ends with // on a line by itself. The identifier allows backward compatibility as the HMMer software evolves. The closing // allows multiple HMMs to be concatenated into a single file to provide a database of HMMs.

The format for an HMM is divided into two regions. The first region contains text information and miscalleneous parameters in a (roughly) tag-value scheme, akin to EMBL formats. This section is ended by a line beginning with the keyword `HMM`. The second region is of a more fixed format and contains the main model parameters. It is ended by the // that ends the entire definition for a single profile-HMM.

Both regions contain probabilities that are used parameterize the HMM. These are stored as integers which are related to the probability via a log-odds calculation. The log-odds score calculation is defined in `mathsupport.c` and is:

$$\text{score} = (\texttt{int})\texttt{floor}(0.5 + (\texttt{INTSCALE} * \log_2(\text{prob}/\text{null-prob})))$$

so conversely, to get a probability from the scores in an HMM save file:

$$\text{prob} = \text{null-prob} * 2^{\text{score}/\texttt{INTSCALE}}$$

`INTSCALE` is defined in `config.h` as 1000.

Notice that you must know a null model probability to convert scores back to HMM probabilities.

The special case of prob = 0 is translated to "*", so a score of * is read as a probability of 0. Null model probabilities are not allowed to be 0.

This log-odds format has been chosen because it has a better dynamic range than storing probabilities as ASCII text, and because the numbers are more meaningful to a human reader to a certain extent: positive values means a better than expected probability, and negative values a worse than expected probability. However, because of the conversion from probabilities, it should be noted that *you should not edit the numbers in a HMMER save file directly*. The HMM is a probabilistic model and expects state transition and symbol emission probability distributions to sum to one. If you want to edit the HMM, you must understand the underlying Plan7 probabilistic model, and ensure the correct summations yourself.

A more detailed description of the format follows.

## Header section

In the header section, each line after the initial identifier has a unique tag of five characters or less. For shorter tags, the remainder of the five characters is padded with spaces. Therefore the first six characters of these lines are reserved for the tag and a space. The remainder of the line starts at the seventh character. The parser does require this.

**HMMER2.0** File format version; a unique identifier for this save file format. Used for backwards compatibility. *Not* necessarily the version number of the HMMER software that generated it; rather, the version number of the last HMMER that changed the format so much that a whole new function had to be introduced to do the parsing. (i.e., HMMER 2.8 might still be writing save files that are headed `HMMER2.0`). **Mandatory.**

**NAME <s>** Model name; `<s>` is a single word name for the HMM. No spaces or tabs may occur in the name. `hmmbuild` will use the `#=ID` line from a SELEX alignment file to set the name. If this is not present, or the alignment is not in SELEX format, `hmmbuild` sets the HMM name using the name of the alignment file, after removing any file type suffix. For example, an HMM built from the alignment file `rrm.slx` would be named `rrm` by default. **Mandatory.**

**ACC <s>** Accession number; `<s>` is a one-word accession number for an HMM. Used in Pfam maintenance. Accessions are stable identifiers for Pfam models, whereas names may change from release to release. Added in v2.1.1. **Optional.**

**DESC <s>** Description line; `<s>` is a one-line description of the HMM. `hmmbuild` will use the `#=DE` line from a SELEX alignment file to set the description line. If this is not present, or the alignment is not in SELEX format, the description line is left blank; one can be added manually (or by Perl script) if you wish. **Optional.**

**LENG <d>** Model length; `<d>`, a positive nonzero integer, is the number of match states in the model. **Mandatory.**

**ALPH <s>** Symbol alphabet; `<s>` must be either `Amino` or `Nucleic`. This determines the symbol alphabet and the size of the symbol emission probability distributions. If `Amino`, the alphabet size is set to 20 and the symbol alphabet to "ACDEFGHIKLMNPQRSTVWY" (alphabetic order). If `Nucleic`, the alphabet size is set to 4 and the symbol alphabet to "ACGT". Case insensitive. **Mandatory.**

**RF <s>** Reference annotation flag; `<s>` must be either `no` or `yes` (case insensitive). If set to `yes`, a character of reference annotation is read for each match state/consensus column in the main section of the file (see below); else this data field will be ignored. Reference annotation lines are currently somewhat inconsistently used. The only major use in HMMER is to specify which columns of an alignment get turned into match states when using the `hmmbuild --hand` manual model construction option. Reference annotation can only be picked up from SELEX format alignments. See description of SELEX format for more details on reference annotation lines. **Optional**; assumed to be no if not present.

**CS <s>** Consensus structure annotation flag; `<s>` must be either `no` or `yes` (case insensitive). If set to `yes`, a character of consensus structure annotation is read for each match state/consensus column in the main section of the file (see below); else this data field will be ignored. Consensus structure annotation lines are currently somewhat inconsistently used. Consensus structure annotation can only be picked up from SELEX format alignments. See description of SELEX format for more details on consensus structure annotation lines. **Optional**; assumed to be no if not present.

**MAP <s>** Map annotation flag; `<s>` must be either `no` or `yes` (case insensitive). If set to `yes`, each line of data for the match state/consensus column in the main section of the file is followed by an extra number. This number gives the index of the alignment column that the match state was made from. This information provides a "map" of the match states ($1..M$) onto the columns of the alignment ($1..alen$). It is used for quickly aligning the model back to the original alignment, e.g. when using `hmmalign --mapali`. Added in v2.0.1. **Optional**; assumed to be no if not present.

**COM <s>** Command line log; `<s>` is a one-line command. There may be more than one `COM` line per save file. These lines record the command line for every HMMER command that modifies the save file. This helps us automatically log Pfam construction strategies, for example. **Optional.**

**CKSUM <d>** Training alignment checksum; `<d>` is a nonzero positive integer. This number is calculated from the training alignment and stored when `hmmbuild` is used. It is used in conjunction with the alignment map information to to verify that some alignment is indeed the alignment that the map is for. Added in v2.0.1. **Optional.**

**GA <f> <f>** Pfam gathering thresholds GA1 and GA2. This is a feature in progress. See Pfam documentation of GA lines. Added in v2.1.1. **Optional.**

**TC <f> <f>** Pfam trusted cutoffs TC1 and TC2. This is a feature in progress. See Pfam documentation of TC lines. Added in v2.1.1. **Optional.**

**NC <f> <f>** Pfam noise cutoffs NC1 and NC2. This is a feature in progress. See Pfam documentation of NC lines. Added in v2.1.1. **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.**

**DATE <s>** Creation date; `<s>` is a date string. This field is only used for logging purposes. **Optional.**

**XT <d>*8** Eight "special" transitions for controlling parts of the algorithm-specific parts of the Plan7 model. The null probability used to convert these back to model probabilities is 1.0. The order of the eight fields is N $\rightarrow$ B, N $\rightarrow$ N, E $\rightarrow$ C, E $\rightarrow$ J, C $\rightarrow$ T, C $\rightarrow$ C, J $\rightarrow$ B, J $\rightarrow$ J. (Another way to view the order is as four transition probability distributions for N,E,C,J; each distribution has two probabilities, the first one for "moving" and the second one for "looping".) For an explanation of these special transitions (and definition of the state names), read the Plan7 architecture documentation. **Mandatory.**

**NULT <d> <d>** The transition probability distribution for the null model (single G state). The null probability used to convert these back to model probabilities is 1.0. The order is G $\rightarrow$ G, G $\rightarrow$ F. **Mandatory.**

**NULE <d>*K** The symbol emission probability distribution for the null model (G state); consists of $K$ (e.g. 4 or 20) integers. The null probability used to convert these back to model probabilities is $1/K$. (Yes, it's a little weird to have a "null probability" for the null model symbol emission probabilities; this is strictly an aesthetic decision, so one can look at the null model and easily tell which amino acids are more common than chance expectation in the background distribution.) **Mandatory.**

**EVD <f> <f>** The extreme value distribution parameters $\mu$ and $\lambda$, respectively; both floating point values. $\lambda$ is positive and nonzero. These values are set when the model is calibrated with `hmmcalibrate`. They are used to determine E-values of bit scores. If this line is not present, E-values are calculated using a conservative analytic upper bound. **Optional.**

**HMM** HMM flag line; flags the end of the header section. Otherwised not parsed. Strictly for human readability, the symbol alphabet is also shown on this line, aligned to the `NULE` fields and the fields of the match and insert symbol emission distributions in the main model. The immediately next line is also an unparsed human annotation line: column headers for the state transition probability fields in the main model section that follows. Both lines are **mandatory.**

## Main model section

All the remaining fields are **mandatory**, except for the alignment map.

The first line in the main model section is atypical; it contains three fields, for transitions from the B state into the first node of the model. *The only purpose of this line is to set the B → D transition probability.* The first field is the score for $1 - t(B \rightarrow D)$. The second field is always "*" (there is no B → I transition). The third field is the score for $t(B \rightarrow D)$. The null probability used for converting these scores back to probabilities is 1.0. In principle, only the third number is needed to obtain $t(B \rightarrow D)$. In practice, HMMER reads both the first and the third number, converts them to probabilities, and renormalizes the distribution to obtain $t(B \rightarrow D)$. [*]

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:

**Match emission line** The first field is the node number (1..M). The HMMER parser verifies this number as a consistency check (it expects the nodes to come in order). The next $K$ numbers for match emission scores, one per symbol. The null probability used to convert them to probabilities is the relevant null model emission probability calculated from the `NULE` line.

If `MAP` was `yes`, then there is one more number on this line, representing the alignment column index for this match state. See `MAP` above for more information about the alignment map, and also see the man pages for `hmmalign --mapali`. Added in v2.0.1. This field is optional, for backwards compatibility with 2.0.

**Insert emission line** The first field is a character of reference annotation (RF), or "-" if there is no reference annotation. The remaining fields are $K$ numbers for insert emission scores, one per symbol, in alphabetic order. The null probability used to convert them to probabilities is the relevant null model emission probability calculated from the `NULE` line.

[*] OK, it's more complicated than it has to be. Sometimes you only discover your temporary insanity when you're trying to document it.

**State transition line** The first field is a character of consensus structure annotation (CS), or "-" if there is no consensus structure annotation. The remaining 9 fields are state transition scores. The null probability used to convert them back from log odds scores to probabilities is 1.0. The order of these scores is given by the annotation line at the top of the main section: it is M $\rightarrow$ M, M $\rightarrow$ I, M $\rightarrow$ D; I $\rightarrow$ M, I $\rightarrow$ D; D $\rightarrow$ M, D $\rightarrow$ D; B $\rightarrow$ M; M $\rightarrow$ E.

The insert emission and state transition lines for the final node $M$ are special. Node $M$ has no insert state, so all the insert emissions are given as "*". (In fact, this line is skipped by the parser, except for its RF annotation.) There is also no next node, so only the B $\rightarrow$ M and M $\rightarrow$ E transitions are valid; the first seven transitions are always "*". (Incidentally, the M $\rightarrow$ E transition score for the last node is always 0, because this probability has to be 1.0.)

Finally, the last line of the format is the "//" record separator.

## Renormalization

After the parser reads the file and converts the scores back to probabilities, it renormalizes the probability distributions to sum to 1.0 to eliminate minor rounding/conversion/numerical imprecision errors. If you're trying to emulate HMMER save files, it might be useful to know what HMMER considers to be a probability distribution. See `Plan7Renormalize()` in `plan7.c` for the relevant function.

**null emissions** The $K$ symbol emissions given on the `NULE` line.

**null transitions** The two null model transitions given on the `NULT` line.

**N,E,C,J specials** Each of the four special states N,E,C,J have two state transition probabilities (move and loop). All four distributions are specified on the `XT` line.

**B transitions** $M$ B $\rightarrow$ M entry probabilities are given by the 9th field in the state transition line of each of the $M$ nodes. The B $\rightarrow$ D transition (from the atypical first line of the main model section) is also part of this state transition distribution.

**match transitions** One distribution of 4 numbers per node; $M \rightarrow M$, $M \rightarrow I$, $M \rightarrow D$, and $M \rightarrow E$ (fields 2, 3, 4, and 10 in the state transition line of each node). Note the asymmetry between B $\rightarrow$ M and M $\rightarrow$ E; entries are a probability distribution of their own, while exits are not.

**insert transitions** One distribution of 2 numbers per node; $I \rightarrow M$, $I \rightarrow I$ (fields 5 and 6 of the state transition line of each node).

**delete transitions** One distribution of 2 numbers per node; $D \rightarrow M$, $D \rightarrow D$ (fields 7 and 8 of the state transition line of each node).

**match emissions** One distribution of $K$ numbers per node; the $K$ match symbol emissions given on the first line of each node in the main section.

**insert emissions** One distribution of $K$ numbers per node; the $K$ insert symbol emissions given on the second line of each node in the main section.

### Note to developers

Though I make an effort to keep this documentation up to date, it may lag behind the code. For definitive answers, please check the parsing code in hmmio.c. The relevant function to see what's being written is WriteAscHMM(). The relevant function to see how it's being parsed is read_asc20hmm().

## 4.2 HMMER null model files

A "null model" is used to calculate HMMER log odds scores. The null model states the expected background occurrence frequencies of the 20 amino acids or the 4 nucleotide bases. The null model also contains a parameter called p1, which is the $G \rightarrow G$ transition probability in the Plan7 null model (see the figure in the Introduction).

For protein models, by default, the 20 residue frequencies are set to the amino acid composition of SWISS-PROT 34, and p1 is set to 350/351 (which, because the Plan7 null model implies a geometric length distribution, states that the mean length of a protein is about 350 residues). For DNA/RNA models, by default, the 4 residue frequencies are set to 0.25 each, and p1 is set to 1000/1001. [In the code, see prior.c:P7DefaultNullModel(), and the amino acid frequencies are set in iupac.c:aafq.]

Each HMM carries its own null model (see above, HMM file format). The null model is determined when the model is built using hmmbuild. The default null model can be overridden using the --null <f> option to hmmbuild, where <f> is the name of a null model file.

Two example null model files, amino.null and nucleic.null, are provided in the Demos subdirectory of the HMMER distribution. (They are copies of the internal default HMMER null model settings.) nucleic.null looks like this:

```
# nucleic.null
#
# Example of a null model file for DNA/RNA sequences.
# The values in this file are the HMMER 2 default
# settings.

Nucleic
0.25     # A
0.25     # C
0.25     # G
0.25     # T
0.999001 # p1
```

Anything on a line following a # is a comment, and is ignored by the software. Blank lines are also ignored. Valid fields are separated by blanks or new lines. Only the order that the fields occur in the file is important, not how they're put on lines; for example, 20 required fields can all occur on one line separated by blanks, or on 20 separate lines.

There must be 6 or 22 non-comment fields in a null model file, occurring in the following order:

**Alphabet type** The first (non-comment) word in the file must be Nucleic or Amino, specifying what kind of sequence the null model is for.

**Emission probabilities** 4 or 20 background frequencies for the amino acids or nucleotides. These *must* come in alphabetical order (the A, C, G, T comments in the example above are only for easier human viewing, and aren't parsed by the software).

**p1 probability** The $G \rightarrow G$ transition probability in the null model. Basically, if the expected mean length of target sequences is $x$, p1 should be $\frac{x}{x+1}$.

Null model files are parsed in `prior.c:P7ReadNullModel()`.

## 4.3 HMMER prior files

Observed counts of emissions (residues) and transitions (insertions and deletions) in a multiple alignment are combined with *Dirichlet priors* to convert them to probabilities in an HMM.

For protein models, by default, HMMER uses a nine-component mixture Dirichlet prior for match emissions, and single component Dirichlet priors for insert emissions and transitions. The nine-component match emission mixture Dirichlet comes from the work of Kimmen Sjölander (Sjölander et al., 1996).

For DNA/RNA models, by default, HMMER uses single component Dirichlets.

Two example null model files, `amino.pri` and `nucleic.pri`, are provided in the `Demos` subdirectory of the HMMER distribution. (They are copies of the internal default HMMER prior settings.)

The way the format of these files is parsed is identical to null models: everything after a # on a line is a comment, the order of occurrence of the fields is important, and fields must be separated by either blanks or newlines.

A prior file consists of the following fields:

**Strategy** Must be the keyword `Dirichlet`. Currently this is the only available prior strategy in the public HMMER release.

**Alphabet type** Must be either `Amino` or `Nucleic`.

**Transition priors** 1 + 8a fields, where a is the number of transition mixture components. The first field is the number of transition prior components, a (often just 1). Then, for each component, eight fields follow: the prior probability of that mixture component (1.0 if there is only one component), then the Dirichlet alpha parameters for the seven transitions, in order of $M \rightarrow M$, $M \rightarrow I$, $M \rightarrow D$, $I \rightarrow M$, $I \rightarrow I$, $D \rightarrow M$, $D \rightarrow I$.

**Match emission priors** 1 + (5 or 21)b fields, where b is the number of match emission mixture components. The first field is the number of match emission mixture components, b. Then, for each component, 5 or 21 fields follows: the prior probability of that mixture component (1.0 if there is only one component), then the Dirichlet alpha parameters for the 4 or 20 residue types, in alphabetical order.

**Insert emission priors** 1 + (5 or 21)c fields, where c is the number of insert emission mixture components. The first field is the number of insert emission mixture components, c. Then, for each component, 5 or 21 fields follows: the prior probability of that mixture component (1.0 if there is only one component), then the Dirichlet alpha parameters for the 4 or 20 residue types, in alphabetical order.

In the code, prior files are parsed by `prior.c:P7ReadPrior()`.

## 4.4 Sequence files

### Supported file formats

HMMER can automatically recognize and parse a number of common file formats. The best supported of these formats are listed below. If you know your sequence database is in one of these formats, you can use the file. If you are formatting sequences yourself, see the section of FASTA format below for unaligned sequences, and the section on SELEX format for alignments; these are the simplest formats to hand type

**Unaligned sequence formats**

| | |
|---|---|
| FASTA | Pearson FASTA format; BLAST databases |
| SWISS-PROT | SWISS-PROT protein sequence database |
| PIR | PIR protein sequence database |
| EMBL | EMBL DNA sequence database |
| GenBank | GenBank DNA database flat files |
| GCGdata | Wisconsin GCG sequence database format |
| GCG | Wisconsin GCG single sequence format |

**Multiple sequence alignment formats**

| | |
|---|---|
| SELEX | The preferred format for HMMER/Pfam (see below) |
| MSF | GCG alignment format |
| CLUSTAL | CLUSTALW and CLUSTALV format |
| A2M | "Aligned FASTA" |

For programs that do sequential, database-style access (i.e. where you'd usually use an unaligned flat file), the alignment formats are read as if they were multiple unaligned sequences.

There is no provision for enforcing that single unaligned sequence formats (i.e. GCG unaligned sequence format) are single sequence only. HMMER will happily try to read more than one sequence if your file contains more than one. However, this may not give the results you expected.

Staden "experiment file" format is parsed using the EMBL file parser, but this functionality is relatively unsupported. There is one wrinkle in this. Staden experiment files use '-' characters to indicate 'N' – i.e., that a base is present in a sequence read, but its identity is unknown. Therefore, the software replaces any '-' in an EMBL sequence with an 'N'. Sometimes people use the unaligned formats to distribute aligned sequences by including gap characters. If EMBL files are used i this way for aligned strings, they must use a different character than '-' to indicate gaps.

### FASTA unaligned sequence format

An example of a simple FASTA file:

```
> seq1 This is the description of my first sequence.
AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCA
CGACGTAGATGCTAGCTGACTCGATGC
> seq2 This is a description of my second sequence.
CGATCGATCGTACGTCGACTGATCGTAGCTACGTCGTACGTAG
CATCGTCAGTTACTGCATGCTCG
```

FASTA is probably the simplest of formats for unaligned sequences. FASTA files are easily created in a text editor. Each sequence is preceded by a line starting with >. The first word on this line is the name of the sequence. The rest of the line is a description of the sequence (free format). The remaining lines contain the sequence itself. You can put as many letters on a sequence line as you want.

Blank lines in a FASTA file are ignored, and so are spaces or other gap symbols (dashes, underscores, periods) in a sequence. Any other non-amino or non-nucleic acid symbols in the sequence should produce an appropriately strident string of warnings on yur terminal screen when you try to use the file.

HMMER currently has a limit of 4095 characters on lines in files. You may see this limitation in NCBI NR databases, where some description lines will be truncated by HMMER.

### SELEX alignment format

An example of a simple SELEX alignment file:

```
# Example selex file

seq1     ACGACGACGACG.
seq2     ..GGGAAAGG.GA
seq3     UUU..AAAUUU.A

seq1  ..ACG
seq2  AAGGG
seq3  AA...UUU
```

SELEX is an interleaved multiple alignment format that arose as an intuitive format that was easy to write and manipulate manually with a text editor like emacs. It is usually easy to convert other alignment formats into SELEX format, even with a couple of lines of Perl, but it can be harder to go the other way, since SELEX is more free-format than other alignment formats. For instance, GCG's MSF format and the output of the CLUSTALV multiple alignment program are similar interleaved formats that can be converted to SELEX just by stripping a small number of non-sequence lines out. Because SELEX evolved to accomodate different user input styles, it is very tolerant of various inconsistencies such as different gap symbols, varying line lengths, etc.

Each line contains a name, followed by the aligned sequence. A space, dash, underscore, or period denotes a gap. If the alignment is too long to fit on one line, the alignment is split into multiple blocks, separated by blank lines. The number of sequences, their order, and their names must be the same in every block (even if a sequence has no residues in a given block!) Other blank lines are ignored. You can add comments to the file on lines starting with a `#`.

SELEX stands for "Systematic Evolution of Ligands by Exponential Enrichment" – it refers to the Tuerk and Gold technology for evolving families of small RNAs for particular functions (Tuerk and Gold, 1990). SELEX files were what we used to keep track of alignments of these small RNA families. It's an interesting piece of historical baggage – most of my lab works on RNA, not protein.

As the format evolved, more features have been added. To maintain compatibility with past alignment files, the new features are added using a reserved comment style. You don't have to worry about adding these extra information lines. They are generally the province of automated SELEX-generating software, such as my `koala` sequence alignment editor or the `COVE` and HMMER sequence analysis packages. This extra information 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.

#### Detailed specification of a SELEX file

1. Any line beginning with a `#=` as the first two characters is a machine comment. `#=` comments are reserved for parsed data about the alignment. Often these features are maintained by software such

as the `koala` editor or the HMMER software, not by hand. The format of `#=` lines is usually quite specific, since they must be parsed by the file-reading software.

2. All other lines beginning with a `%` or `#` as the first character are user comments. User comments are ignored by all software. Anything may appear on these lines. Any number of comments may be included in a SELEX file, and at any point.
3. Lines of data consist of a name followed by a sequence. The total length of the line must be smaller than 4096 characters.
4. Names must be a single word. Any non-whitespace characters are accepted. No spaces are tolerated in names: names MUST be a single word. Names must be less than 32 characters long.
5. In the sequence, any of the characters `-_.` or a space are recognized as gaps. Any other characters are interpreted as sequence. Sequence is case-sensitive. There is a common assumption by my software that upper-case symbols are used for consensus (match) positions and lower-case symbols are used for inserts. This language of "match" versus "insert" comes from the hidden Markov model formalism (Krogh et al., 1994). To almost all of my software, this isn't important, and it immediately converts the sequence to all upper-case after it's read.
6. Multiple different sequences are grouped in a block of data lines. Blocks are separated by blank lines. No blank lines are tolerated between the sequence lines in a block. Each block in a multi-block file of a long alignment must have its sequences in the same order in each block. The names are checked to verify that this is the case; if not, only a warning is generated. (In manually constructed files, some users may wish to use shorthand names in subsequent blocks after an initial block with full names – but this isn't recommended.)

### Optional SELEX file annotation

### Header

Several pieces of information can be added to the header of the file using the following machine comments:

**`#=ID <s>`** Name of the alignment; `<s>` is one word, no spaces (example: "rrm"). If present, HMMER's `hmmbuild` picks this up and uses it as the name of an HMM.

**`#=AC <s>`** Accession number of the alignment; `<s>` is one word, no spaces (example: "PF99999"). If present, HMMER's `hmmbuild` picks this up and uses it as the accession number of an HMM. (This is fairly specific to the support of the Pfam HMM database.)

**`#=DE <s>`** Description line; `<s>` is a one-line description of the alignment. If present, HMMER's `hmmbuild` picks this up and uses it as the description line of an HMM.

**`#=AU <s>`** Author line; `<s>` is a one-line description of who made the alignment. HMMER doesn't use this line for anything.

**`#=GA <f1> <f2>`** Gathering cutoffs; `<f1>` is the per-sequence GA1 cutoff, `<f2>` is the per-domain GA2 cutoff. Both numbers are floating point numbers (HMMER bit scores). This is very specific to Pfam support. The GA cutoffs specify the HMMER score cutoffs used in automatically generating the Pfam full alignments.

**#=TC <f1> <f2>** Trusted cutoffs; <f1> is the per-sequence TC1 cutoff, <f2> is the per-domain TC2 cutoff. Both numbers are floating point numbers (HMMER bit scores). This is very specific to Pfam support. The TC cutoffs specify the HMMER scores of the lowest-scoring sequence/domain included in a Pfam full alignment.

**#=NC <f1> <f2>** Noise cutoffs; <f1> is the per-sequence NC1 cutoff, <f2> is the per-domain NC2 cutoff. Both numbers are floating point numbers (HMMER bit scores). This is very specific to Pfam support. The NC cutoffs specify the HMMER scores of the highest scoring sequence/domain that were not included in a Pfam full alignment.

**Sequence header**

Additional per-sequence information can be placed after the main header lines and before any blocks appear. These lines, one per sequence and in exactly the same order as the sequences appear in the alignment, are formatted like

```
#=SQ <name> <wgt> <source> <accession> <start..stop::orig length> <desc>
```

This information includes a sequence weight (for compensating for biased representation of subfamilies of sequences in the alignment); source information, if the sequence came from a database, consisting of identifier, accession number, and source coordinates; and a description of the sequence.

If a #=SQ line is present, all the fields must be present on each line and one #=SQ line must be present for each sequence. If no information is available for a field, use '-' for all the fields except the source coordinates, which would be given as '0'.

**Secondary structure annotation**

Lines beginning with #=SS or #=CS are individual or consensus secondary structure data, respectively. #=SS individual secondary structure lines must immediately follow the sequence they are associated with. There can only be one #=SS per sequence. All, some, or none of the sequences may have #=SS annotation.

#=CS consensus secondary structure predictions precede all the sequences in each block. There can only be one #=CS per file.

I use one-letter codes to indicate secondary structures. Secondary structure strings are aligned to sequence blocks just like additional sequences.

For RNA secondary structure, the symbols > and < are used for base pairs (pairs point at each other). + indicate definitely single-stranded positions, and any gap symbol indicates unassigned bases or single-stranded positions. This description roughly follows (Konings and Hogeweg, 1989). For protein secondary structure, I use E to indicate residues in $\beta$-sheet, H for those in $\alpha$-helix, L for those in loops, and any gap symbol for unassigned or unstructured residues.

RNA pseudoknots are represented by alphabetic characters, with upper case letters representing the 5' side of the helix and lower case letters representing the 3' side. Note that this restricts the annotation to a maximum of 26 pseudoknots per sequence.

**Reference coordinate system**

Alignments are usually numbered by some reference coordinate system, often a canonical molecule. For instance, tRNA positions are numbered by reference to the positions of yeast tRNA-Phe.

A line beginning with #=RF preceding the sequences in a block gives a reference coordinate system. Any non-gap symbol in the #=RF line indicates that sequence positions in its columns are numbered. For instance, the #=RF lines for a tRNA alignment would have 76 non-gap symbols for the canonical numbered columns; they might be the aligned tRNA-Phe sequence itself, or they might be just X's.

The #=RF line also provides a good way to mark conserved key residues in a protein alignment. HMMER picks up both the #=RF and #=CS lines when it builds an HMM, if they're there, and will display them in the context of any alignments to target sequences.

### An example SELEX file with annotation

Here's an example SELEX file with examples of all the optional fields. The sequences are evolved RNA ligands for the phage R17 coat protein (Schneider et al., 1992).

```
#=ID r17
#=AC RNA00001
#=DE RNA ligands evolved to bind phage R17 coat protein
#=GA 0.0 0.0
#=TC 0.0 0.0
#=NC 0.0 0.0
#=AU COVE 2.2.3 Wed Dec  2 08:39:58 1998

#=SQ lig28 1.0 - - 1..29:29 R17 coat protein ligand 28
#=SQ lig1  1.0 - - 1..17:17 R17 coat protein ligand 1
#=SQ lig2  1.0 - - 1..29:29 R17 coat protein ligand 2

#=RF         ***A** loop *****
#=CS  .......>>>+>> ++++ <<<<<.......
lig28 GGAGUAAGAUAGC AUCA GCAUCUUGUUCC
#=SS   >>>>>>>>>+>> ++++ <<<<<<<<<<<<
lig1        GUUCACC AUCA GGGGAC
#=SS        >>>>+>> ++++ <<<<<<
lig2  AUGGAUGCGCACC AUCA GGGCGUAUCUAU
#=SS  >>>>>>>>>>+>> ++++ <<<<<<<<<<<<
```

# Chapter 5

# Installation

## 5.1 System requirements and portability

HMMER is designed to run on UNIX platforms. The code is POSIX-compliant ANSI C. You need a UNIX machine and an ANSI C compiler to build and use it.

Some optional tests and utilities use Perl. `make check` will only work if you have Perl installed. However, Perl isn't necessary to make HMMER work.

HMMER includes support for two kinds of parallelization: POSIX threads and PVM (Parallel Virtual Machine). Most modern UNIX OS's support POSIX threads. HMMER's configure script attempts to automatically detect your threads library, if you have one, and will build in multithreading by default – so you don't need to do anything special to get multithreading. PVM, in contrast, is a separate, third-party software application, and you will have to install and configure it before building HMMER with PVM support. For this reason, PVM support in HMMER is not enabled by default. See the PVM section of this chapter for more details.

HMMER 2 should be easy to port to non-UNIX operating systems, provided they support ANSI C and some reasonable level of POSIX compliance. A WinNT distribution is available by anonymous FTP from Time Logic, Inc. HMMER 1 was ported by other people to Digital VAX/VMS, Apple MacOS, Win95, and WinNT with relatively little difficulty, and I've made efforts to improve the portability of the HMMER 2 code since then. I would greatly appreciate receiving diffs for any ports of HMMER to any platform.

## 5.2 Installing a precompiled distribution

Precompiled binary distributions of HMMER are available for at least the following platforms:

- Intel PC/Linux
- Intel PC/FreeBSD
- Intel PC/Sun Solaris 2.x
- Silicon Graphics/IRIX 5.x and 6.x
- Sun Sparc/ Solaris 2.x
- DEC Alpha/Digital UNIX 4.0

To install a precompiled distribution, do the following:

1. Download the distribution from `http://hmmer.wustl.edu/`.

2. Uncompress and un-tar it; for example,
   > **uncompress hmmer-intel-linux.tar.Z**
   > **tar xf hmmer-intel-linux.tar**
   A new directory `hmmer-xx` is created, where "xx" is the HMMER version number.

3. Go to the top level directory and run the configure script.
   > **cd hmmer-xx**
   > **./configure**

4. Edit the Makefile in the top level directory.

   To run HMMER in the distribution directory, you don't need to edit the Makefile at all. But to permanently install HMMER on your system, set the make variables BINDIR and MANDIR to be the directories where you want HMMER executables and man pages to be installed. If you are installing in directories `/usr/local/bin/` and `/usr/local/man/man1/`, you don't need to change anything.

5. Type `make install` to install the programs and man pages. You may have to become root, depending on where you're installing.

6. Type `make clean` to clean up.

## 5.3 Compiling from a source-only distribution

1. Download the distribution from `http://hmmer.wustl.edu/`.

2. Uncompress and un-tar it:
   > **uncompress hmmer.tar.Z**
   > **tar xf hmmer.tar**
   A new directory `hmmer-xx` is created, where "xx" is the HMMER version number.

3. Go to the top level directory and run the configure script.
   > **cd hmmer-xx**
   > **./configure**
   If you want to include the optional PVM support, do:
   > **./configure --with-pvm**
   If you include PVM, the system that you're compiling on must already be set up for PVM; specifically, the environment variables PVM_ROOT and PVM_ARCH must be set, so HMMER can find the PVM headers and library.

   If you want to disable POSIX threads support, do:
   > **./configure --disable-threads**

   If you want to change the choice of compilation flags (CFLAGS) or the compiler (CC), set these as environment variables:
   > **env CC=gcc CFLAGS="-O6" ./configure**
   For other generic configuration options, see the documentation for GNU autoconf 2.12. – but it may be easier to just edit the Makefile (see below).

4. Edit the top of the Makefile.

   To build and test HMMER in its source directory, you don't need to edit the Makefile at all.

   To permanently install HMMER on your system, set the make variables BINDIR and MANDIR to be the directories where you want HMMER executables and man pages to be installed. If you are installing the programs in `/usr/local/bin` and the man pages in `/usr/local/man/man1`, you don't need to change anything.

   The default Makefile gets configured into a reliable but not necessarily optimal choice of compiler and compilation flags. The package is known to build "out of the box" on SGI IRIX, Sun Solaris, GNU/Linux, FreeBSD, or DEC Digital Unix platforms without any special modifications. However, you may want to change the CC or CFLAGS variables to suit your system. In particular, you can play with the compiler options in CFLAGS to try to get more speed, if you're compiler-fluent. The Makefile includes some hints for various platforms.

   On SunOS 4.1.x systems, you will have to use the GNU gcc compiler, because SunOS cc is not ANSI-compliant.

5. Type `make` to build the programs.

6. (Optional) Type `make check` to compile and run a test suite. Some of the tests require that you have Perl installed. None of these tests should fail.

7. Type `make install` to install the programs and man pages. You may have to become root, depending on where you're installing.

8. Type `make clean` to clean up.

## 5.4 Environment variable configuration

These steps are optional, and will only apply (I think) to sufficiently POSIX-compliant operating systems like UNIX, Linux, or WinNT.

HMMER reads four environment variables, and is designed to coexist peacefully with an installation of WUBLAST or NCBI BLAST:

**BLASTDB** - directory location of FASTA-formatted sequence databases

**BLASTMAT** - directory location of BLAST substitution matrices

**HMMERDB** - directory location of HMM databases (e.g. PFAM)

**HMMER_NCPU** - maximum number of CPUs to utilize in a multiprocessor

If you have installed BLAST, you probably already have the two BLAST environment variables set in system-wide or user-specific .cshrc files.

All four variables are optional. If they are set up, you can simplify command lines to:

> **hmmpfam pfam my.query**
> **hmmsearch my.hmm swiss35**

instead of

> **hmmpfam /some/long/path/to/databases/pfam my.query**
> **hmmsearch my.hmm /some/long/path/to/databases/swiss35**

## 5.5 Parallelization using threads

HMMER includes support for two kinds of parallelization: POSIX threads and PVM (Parallel Virtual Machine). Threads support is built in by default; PVM is not.

If you have a multi-processor UNIX machine, odds are that you have a POSIX threads library. This kind of parallelization is easy from the standpoint of a HMMER user. HMMER will happily use all your processors (fewer, if you wish). The binary distributions of HMMER all come with multithreading capability built in.

To *disable* threads support, add `--disable-threads` to the command line for `./configure` before you compile a source distribution. Multithreading is completely optional, and the software will work fine without it.

Like multithreaded BLAST, a multithreaded HMMER search will use all the available CPUs. Sometimes this is not desired; you may want HMMER searches to leave some spare processing power for other work. The environment variable HMMER_NCPU sets the maximum number of CPUs that any HMMER job will use. You can also set a limit on an individual process using the `--cpu <n>` option.

## 5.6 Parallelization using PVM

Parallelization across multiple machines (as opposed to multithreading on a single multiprocessor machine) can be done with PVM, the Parallel Virtual Machine software from Oak Ridge National Labs.

PVM is freely available. You can obtain it from `http://www.epm.ornl.gov/`. You must install and configure PVM before compiling PVM support into HMMER. During compilation, HMMER needs to see the environment variables PVM_ROOT and PVM_ARCH, and the PVM header files and libraries must be found in the appropriate place under $PVM_ROOT.

To enable PVM support in HMMER, add `--with-pvm` to the command line for `./configure` before you compile a source distribution. PVM is completely optional, and the software will work fine without it.

Whereas multithreading requires no special configuration once support is compiled into HMMER, configuring a PVM cluster and using HMMER on it is a little more involved.

### Configuring a PVM cluster for HMMER

Here, I will assume you're already familiar with PVM.

Designate one machine as the "master", and the other machines as "slaves". You will start your HMMER process on the master, and the master will spawn jobs on the slaves using PVM.

Install PVM on the master and all the slaves. On the master, make sure the environment variables PVM_ROOT and PVM_ARCH are set properly (ideally, in a system-wide .cshrc file).

Add the master's name to your .rhosts or /etc/hosts.equiv file on the slaves, so the slaves accept rsh connections from the master.

Put copies of HMMER executables in a directory on the master and all the slaves. For each PVM-capable program (`hmmcalibrate`, `hmmpfam`, and `hmmsearch`, there is a corresponding slave PVM program (`hmmcalibrate-pvm`, `hmmpfam-pvm`, and `hmmsearch-pvm`). The master machine needs copies of all the HMMER programs, including the slave PVM programs. The slaves only need copies of the three slave PVM programs. (You never need to start the slave programs yourself; PVM does that. You just need to make sure they're installed where PVM can see them.)

The PVM implementation of `hmmpfam` needs a copy of any HMM databases you may search to be installed on the master and every slave. All HMM databases must be indexed with `hmmindex`. The reason is that `hmmpfam` is I/O bound; the PVM implementation can't distribute an HMM database fast enough over a typical cluster's Ethernet. Instead, each PVM node accesses its own copy of the HMM database, distributing the I/O load across the nodes. `hmmcalibrate` and `hmmsearch`, in contrast, are freestanding. Only the master node needs to be able to access any HMM and/or sequence files.

Write a PVM hostfile for the cluster. Specify the location of the HMMER executables using the ep= directive. Specify the location of `pvmd` on the slaves using the dx= directive (alternatively, you can make sure PVM_ROOT and PVM_ARCH get set properly on the slaves). For the slaves, use the wd= directive to specify the location of the HMM databases for `hmmpfam` (alternatively, you can make sure HMMERDB gets set properly on the slaves). Use the sp= directive to tell HMMER how many processors each node has (and hence, how many independent PVM processes it should start); sp=1000 means 1 CPU, sp=2000 means 2 CPUs, sp=4000 means 4 CPUS, etc.

Start the PVM by typing
> **pvm hostfile**
(where "hostfile" is the name of your hostfile) on the master. Make sure all the nodes started properly by typing > **conf**
at the PVM console prompt. Type > **quit**
to exit from the PVM console, which leaves the PVM running in the background. You should only need to start PVM once. (We have a PVM running continuously on our network right now, waiting for HMMER jobs.)

Once PVM is running, at any time you can run HMMER programs on the master and exploit your PVM, just by adding the option `--pvm`; for instance,
> **hmmpfam --pvm Pfam my.query**
parallelizes a search of a query sequence in the file my.query against the Pfam database.

Once PVM is properly configured and your slave nodes have the required slave programs (and databases, in the case of `hmmpfam`), the only difference you will notice between the serial and the PVM version is a (potentially massive) increase in search speed. Aside from the addition of the `--pvm` option on the command line, all other options and input/output formats remain identical.

### Example of a PVM cluster

The St. Louis Pfam server runs its searches using HMMER on a PVM cluster called Wulfpack. I'll use it as a specific example of configuring a PVM cluster. It's a little more intricate than you'd usually need for personal use, just because of the details of running PVM jobs in a standalone way from CGI scripts on a Web server.

The master node is the Web server, `fisher`. The slave nodes are eight rack-mounted dual processor Intel/Linux boxes called `wulf01` through `wulf08`. Collectively, we refer to this cluster as Wulfpack; it is a Beowulf-class Linux computing cluster.

PVM 3.3.11 is installed in `/usr/local/pvm3` on the master and the slaves.

On fisher, all HMMER executables are installed in `/usr/local/bin`. On the wulf slave nodes, the three PVM slave executables are installed in `/usr/local/wulfpack`.

Pfam and PfamFrag, two Pfam databases, are installed on the wulf slave nodes in `/usr/local/wulfpack.` They are converted to binary format using `hmmconvert -b`, then indexed using `hmmindex`. (Using binary format databases is a big performance win for `hmmpfam` searches, because `hmmpfam` is I/O bound and binary HMM databases are smaller.)

An `ls` of `/usr/local/wulfpack` on any wulf node looks like:

```
[eddy@wulf01 /home]$ ls /usr/local/wulfpack/
Pfam            PfamFrag         hmmcalibrate-pvm   hmmsearch-pvm
```

```
Pfam.gsi          PfamFrag.gsi     hmmpfam-pvm
```

The PVM hostfile for the cluster looks like:

```
# Config file for Pfam Web server PVM
#
* ep=/usr/local/bin sp=1000
fisher.wustl.edu
* lo=pfam dx=/usr/local/pvm3/lib/pvmd ep=/usr/local/wulfpack sp=2000
wulf01
wulf02
wulf03
wulf04
wulf05
wulf06
wulf07
wulf08
```

Note one wrinkle specific to configuring Web servers: the web server is running HMMER as user "nobody" because it's calling HMMER from a CGI script. We can't configure a shell for "nobody" on the slaves, so we create a dummy user called "pfam" on each wulf node. The lo= directive in the PVM hostfile tells the master to connect to the slaves as user "pfam". On each slave, there is a user "pfam" with a .rhosts that looks like:

```
fisher nobody
fisher.wustl.edu nobody
```

which tells the wulf node to accept rsh connections from fisher's user "nobody".

Also note how we use the sp= directive to tell HMMER (via PVM) that the wulf nodes are dual processors. fisher is actually a dual processor too, but by setting sp=1000, HMMER will only start one PVM process on it (leaving the other CPU free to do all the things that keep Web servers happy).

The trickiest thing is making sure PVM_ROOT and PVM_ARCH get set properly. For my own private PVM use, my `.cshrc` contains the lines:

```
setenv PVM_ROOT    /usr/local/pvm3
setenv PVM_ARCH    `$PVM_ROOT/lib/pvmgetarch`
```

But for the web server PVM, it's a little trickier. We start the Web server PVM as user "nobody" on fisher using a local init script, `/etc/rc.d/init.d/pvm_init`. With its error checking deleted for clarity, this script basically looks like:

```
#!/bin/sh
wulfpack_conf=/home/www/pfam/pfam-3.1/wulfpack.conf
. /usr/local/pvm3/.pvmprofile
$PVM_ROOT/lib/pvmd $wulfpack_conf >/dev/null &
```

We call this at boot time by adding the line `su nobody -c "sh /etc/rc.d/init.d/pvm_init` to our `rc.local` file. .pvmprofile is a little PVM-supplied script that properly sets PVM_ROOT and PVM_ARCH, and wulfpack.conf is our PVM hostfile.

The relevant lines of the CGI Perl script that runs HMMER jobs from the Web server (again, heavily edited for clarity) are:

```
# Configure environment for PVM
$ENV{'HMMERDB'}    = "/usr/local/wulfpack:/home/www/pfam/data/"
$ENV{'PVM_EXPORT'} = "HMMERDB";
$output = `/usr/local/bin/hmmpfam --pvm Pfam /tmp/query`;
```

The trick here is that we export the HMMERDB environment variable via PVM, so the PVM processes on wulf nodes will know where to find their copy of Pfam.

PVM is relatively complex, but with luck, this and the PVM documentation give you enough information to get HMMER running on a cluster. It's well worth it. Wulfpack was simple to assemble; besides the eight rack-mounted machines, there's a terminal switch, a single console, a 10baseT Ethernet switch, and a UPS. Each machine runs stock Red Hat Linux (we don't need no steenking Extreme Linux hype). The whole thing cost us $20K, but it runs HMMER searches as fast as a 16-processor SGI Origin – and it's bigger and has more blinking lights than an Origin, so it's more impressive to look at.

## 5.7 Recommended systems

HMMER is currently developed and maintained on the following three systems:

- Silicon Graphics Origin200 dual R10K server, IRIX 6.5
- NEC Versa 6050MX laptop, Intel Pentium, Red Hat Linux 4.2, configured by VA/Research
- Dell Dimension desktop, Intel PentiumII, Red Hat Linux 4.2

Since these are the development systems, similar systems are the least likely to show portability problems.

At each release, HMMER is also tested on the following platforms: Sun Sparc Solaris, Sun x86 Solaris, DEC Alpha/Digital Unix, and Intel PC/FreeBSD. These systems are also unlikely to show portability problems.

## 5.8 Make targets

There are a variety of make targets in the toplevel Makefile. For completeness, they are summarized here. *You shouldn't need to know this stuff.* Targets marked (Private) are targets which are not supported in the public HMMER release. They only run in my local development environment.

**all** Compiles the source code; puts the compiled programs in the binaries/ subdirectory.

**check** Compiles and runs the testsuite.

**install** Installs the programs in BINDIR and the man pages in MANDIR.

**clean** Cleans up the directory, leaving the distribution files.

**verify** Runs consistency checks on the package. (Private)

**doc** Builds the Userguide from source. (Private)

**dist** Checks out a complete distribution from RCS and assembles the .tar.Z file. (Private)

**ftpdist** Installs a new release on the FTP site. (Private)

**stable** Symlinks a new FTP release to hmmer.tar.Z. (Private)

**almostclean** Like 'make clean', but leaves the binaries/ subdirectory. Used for making an executable distribution. (Private)

# Chapter 6

# License terms

If you're just going to use HMMER to analyze sequence, it's free to you, and you can stop reading the legalese. If you have a specially licensed (non-GPL) copy of HMMER from Washington University, you have your own legalese, in the form of a separate written WashU licensing contract; this chapter does not apply to you. But if you have a public copy of HMMER and you're interested in modifying or borrowing the source code or distributing the software yourself, keep reading.

HMMER is a copyrighted work, and is not public domain software. It is distributed under an "Open Source" license, the GNU General Public License (GPL). In brief, what the license means is that:

- If you're just going to use HMMER to analyze sequence, it's free.
- If you want to redistribute *unmodified* copies of HMMER, you're free to do so. The license even permits you to sell copies of HMMER, or include it (in unmodified form) as part of an otherwise proprietary software package. The license and my copyright must remain unchanged.
- If you want to *modify* HMMER and use your modified copy *internally* (in your company or department), you're free to do so.
- If you want to modify HMMER or borrow any part of its source code for your software package, and you plan to redistribute your code (either freely or for profit), you *must*:
  - freely distribute your code under the GNU GPL, retaining my copyright on the parts I wrote; or
  - if you're a free software developer but you prefer a different open source license, contact me and I'll typically ship you the code under your favorite open source license. For small chunks of code, I will typically ship you source with a very permissive open source license (not the GPL) so you can do what you like with it without worrying about the GPL's viral effects; or
  - if you're a proprietary software developer, contact Washington University to arrange for non-GPL licensing terms. For commercial licenses, we will typically ask for a fee and/or royalties.

## 6.1 "Manifesto"

Because HMMER has commercial value, I spend an inordinate amount of time explaining the choice of the GPL. Open source licenses are widely misunderstood.

Most of my funding comes from the National Human Genome Research Insitute at the National Institutes of Health (NIH). NIH grants policy states that the results of NIH-funded research must be made available to qualified scientists upon request. The free exchange of information is also a fundamental tenet of publicly funded science. The simplest means of disseminating information about how HMMER works is to distribute documented source code. I take this responsibility seriously. While I am in academia, my software will be freely available in source code form.

At the same time, I'm not going to put HMMER in the public domain. If I did, someone could incorporate HMMER into a proprietary software package without my permission or knowledge. I couldn't see the improvements they were making to my work. They might sell the proprietary software and not contribute a fair share of the profits to Washington University or my research lab. So I need a distribution license that lets me distribute it freely, while at the same time trying to make sure that I don't get taken advantage of.

Some developers license their software under restricted "academic only" licenses, such that "commercial" users have to pay license fees. Having worked in a biotech startup, I strongly oppose this sort of license. Most industry scientists contribute to basic research, and they consider themselves no different from their counterparts in academia. The line I want to draw is between open and proprietary, not academic and commercial. Commercial scientists can be open, and academic scientists can be proprietary. (And NIH's free resource distribution policy does not distinguish between academic and industrial peers.)

Open Source licenses (aka "free" software licenses) like the BSD license, the Perl Artistic License, the Netscape Public License, and the GNU Public License do what I need. The GNU Public License is the best known of these licenses. So long as you play the game – that you are working in a nonproprietary manner, accepting that any of your modifications to my code must also be GPL'ed and made freely available as open source code – you get free, open access to HMMER source code. But if you want to use HMMER in a proprietary way, the GPL does not grant you that right, so you must contact the Washington University tech transfer people to arrange a special proprietary license (and, unsurprisingly, we'll charge you fair license fees).

The GPL is often misinterpreted as a socialist license, and I sometimes field questions about why I'm so determined to oppose commercial development, or even undermine it (by distributing free software). This is a misunderstanding. American universities have a responsibility, under the 1980 Bayh/Dole act, to transfer technology to the private sector. I respect the economic goals of Bayh/Dole. In my view, there is a productive tension between Bayh/Dole and scientific ethics. The purpose of the HMMER GPL is not to impede commercial development, but rather to maintain this tension. As an academic, my primary responsibility is to make my work available. As an American research university, the job of Washington University and our technology transfer office is to make HMMER available for proprietary use under appropriate terms. We can and do license HMMER for proprietary use. We're just not going to *give it away* for proprietary use, that's all. If you want to make a profit from HMMER, we want a fair share so we can put it towards the research that develops HMMER.

Whew. You think that was long-winded? That's nothing; the full text of the GNU GPL follows:

**GNU GENERAL PUBLIC LICENSE**
Version 2, June 1991



Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## GNU GENERAL PUBLIC LICENSE
## TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

# Chapter 7

# Acknowledgements and history

> *It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage, than the creation of a new system. For the initiator has the emnity of all who would profit by the preservation of the old institutions and merely lukewarm defenders in those who would gain by the new ones.*
>
> – Niccolo Machiavelli

HMMER 1 was developed at the MRC Laboratory of Molecular Biology, Cambridge UK, while I was a postdoc with Dr. Richard Durbin. I thank the Human Frontier Science Program and the National Institutes of Health for their support.

HMMER 1.8 (and subsequent minor releases) was the first public release of HMMER in April 1995. A number of modifications and improvements went into HMMER 1.9 code, but 1.9 was never released. Some versions of HMMER 1.9 did inadvertently escape St. Louis and make it to other sites, but it was never documented or supported. HMMER 1.9 collapsed under its own weight in 1996, when the number of ugly hacks increased to a critical mass.

HMMER 2 is a nearly complete rewrite, based on a new model architecture dubbed "Plan 7". Implementation was begun in November 1996 at Washington University in St. Louis. I thank the Washington University Dept. of Genetics, the NIH National Human Genome Research Institute, and Monsanto for their support during this (extremely stressful) time. There is nothing like throwing four years of work away and starting fresh to make you question your sanity. Working on a book (Durbin et al., 1998) and starting up a lab at the same time made it all doubly "exciting". If you are so bored as to actually read the code, you will run across inexplicable comments that note where I was when I wrote various parts, making up a sort of disjointed diary of this period; amongst other places, parts of HMMER 2 were written in airport lounges, on TWA flights 720 and 721 to and from London, in Graeme Mitchison's kitchen in Cambridge, and on vacations while my (ex-)wife wasn't watching. I therefore owe special thanks (I think) to 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 MRC-LMB computational molecular biology discussion group has contributed many ideas to HMMER. In particular, I thank Richard Durbin, Graeme Mitchison, Erik Sonnhammer, Alex Bateman, Ewan Birney, Gos Micklem, Tim Hubbard, Roger Sewall, David MacKay, and Cyrus Chothia. Any errors in the code, though, are my fault alone, of course.

Sequence format parsing (`sqio.c`) in HMMER is derived from an early release of the `READSEQ` package by Don Gilbert, Indiana University. Thanks to Don for an excellent piece of software; and apologies for the mangling I've put it through since. The file `hsregex.c` is a derivative of Henry Spencer's regular expression library; thanks, Henry. Several miscellaneous functions in `sre_math.c` are taken from public domain sources and are credited in the code's comments. `masks.c` includes a modified copy of the XNU source code from David States and Jean-Michel Claverie.

In many other places, I've reimplemented algorithms described in the literature. These are too numerous to credit and thank here. The original references are given in the comments of the code. However, I've borrowed (stolen?) more than once from the following folks that I'd like to be sure to thank: Steve Altschul, Pierre Baldi, Phillip Bucher, Warren Gish, David Haussler, Steve and Jorja Henikoff, Richard Hughey, Kevin Karplus, Anders Krogh, Bill Pearson, and Kimmen Sjolander.

HMMER is developed on Silicon Graphics and Linux machines in my lab. Richard Durbin at the Sanger Centre (Hinxton UK) and LaDeana Hillier and Warren Gish at the WashU Genome Sequencing Center have generously donated time on Sun, Intel/Solaris, and DEC Alpha platforms for additional development work. I thank Silicon Graphics, especially Juli Nash and Wade McLain, for their continued support and assistance. Dave Cortesi at SGI contributed much useful advice on the POSIX threads implementation. I am proud to acknowledge a tremendous debt to the development tools that I use from the free software community: an incomplete list includes GNU gcc, gdb, emacs, and autoconf; Cygnus' and others' egcs compiler; Conor Cahill's dbmalloc library; Bruce Perens' ElectricFence; Walter Tichy's RCS; Larry Wall's perl; LaTeX and TeX from Leslie Lamport and Don Knuth; Nikos Drakos' latex2html; Thomas Phelps' PolyglotMan; Linus Torvalds' Linux operating system; and the folks at Red Hat Linux.

Finally, I'd like to cryptically thank Dave "Mr. Frog" Pare and Tom "Chainsaw" Ruschak for a totally unrelated free software product that was historically instrumental in HMMER's development, for reasons that are best not discussed while sober.

# Bibliography


Altschul, S. F. (1991). Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–565.

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.

Barrett, C., Hughey, R., and Karplus, K. (1997). Scoring hidden Markov models. *Comput. Applic. Biosci.*, 13:191–199.

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.

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.

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.

Konings, D. and Hogeweg, P. (1989). Pattern analysis of RNA secondary structure: Similarity and consensus of minimal-energy folding. *J. Mol. Biol.*, 207:597–614.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K., and Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531.

Pearson, W. and Lipman, D. (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.

Schneider, D., Tuerk, C., and Gold, L. (1992). Selection of high affinity RNA ligands to the bacteriophage R17 coat protein. *J. Mol. Biol.*, 228:862–869.

Sjölander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. (1996). Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. *Comput. Applic. Biosci.*, 12:327–345.

Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197.

Sonnhammer, E. L., Eddy, S. R., and Durbin, R. (1997). Pfam: A comprehensive database of protein families based on seed alignments. *Proteins*, 28:405–420.

Sonnhammer, E. L. L., Eddy, S. R., Birney, E., Bateman, A., and Durbin, R. (1998). Pfam: Multiple sequence alignments and HMM-profiles of protein domains. *Nucl. Acids Res.*, 26:320–322.

Taylor, W. R. (1986). Identification of protein sequence homology by consensus template alignment. *J. Mol. Biol.*, 188:233–258.

Tuerk, C. and Gold, L. (1990). Systematic evolution of ligands by exponential enrichment. *Science*, 249:505–510.