

Seqool

v 2.0

A SEQUENCE ANALYSIS TOOL

Program manual

Signal search, pattern recognition, and sequence statistics

2006

Magnus Wang

© Copyright 2006 Magnus Wang

Heidelberg, Germany, 2006

This publication may be reproduced for educational use (i.e. for education and research at public universities) free of charge in any format or medium. The material must be acknowledged as copyright with the title and source of the publication specified. All rights reserved for any uses other than educational, and especially for commercial uses. For non-educational uses no part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by means of electronic mechanical, photocopying recording or otherwise, without the prior permission in writing of the publisher.

Contents

1. Overview	1
2. Installation	1
3. Analysis of a single sequences with <i>Seqool</i>	2
3.1. Basic orientation in <i>Seqool</i>	2
3.2. Opening files	4
3.3. Sequence composition	5
3.3.1. Base composition, GC content, nucleotide/dinucleotide frequencies	5
3.3.2. Codon usage, codon preference	6
3.4. Searching for patterns	8
3.4.1. Text search	8
3.4.2. Stop codon search	11
3.4.3. Searching with pattern recognition models	12
3.5. Additional functions	16
3.5.1. Exporting sequences	16
3.5.2. Creating subsequences	16
3.6. Customizing <i>Seqool</i>	17
3.6.1. Graphical output	17
3.6.2. Configuring <i>Seqool</i>	17
4. Analysis of multiple sequences with <i>SeqoolM</i>	18
4.1. Basic orientation in <i>SeqoolM</i>	18
4.2. Opening files	20
4.3. Sequence composition	22
4.3.1. Base composition, GC content, nucleotide / dinucleotide frequencies	22
4.3.2. Codon usage, codon preference	23
4.3.3. Calculation of oligonucleotide frequencies	25
4.4. Over- and under-represented words	26
4.4.1. Calculation of over-/under-represented <i>words</i>	26
4.4.2. Clustering of over-/under-represented <i>words</i>	28
4.5. Searching for patterns	34
4.5.1. Text search	34
4.5.2. Searching with pattern recognition models	37
4.6. Customizing <i>SeqoolM</i>	41
4.6.1. Configuring <i>SeqoolM</i>	41
4.6.2. Program priority	41
5. Building pattern recognition models	43
5.1. Profiles (WMM, PSSM)	43
5.1.1. Introduction	43
5.1.2. Increased performance using information content	44
5.1.3. Adding pseudocounts	45
5.1.4. Higher order models	45

5.1.5.	Building a WMM/PSSM	46
5.2.	Maximum Dependence Decomposition (MDD).....	47
5.2.1.	Introduction.....	47
5.2.2.	Example.....	47
5.2.3.	Building an MDD model.....	52
5.3.	Profile Hidden Markov Models (PHMM)	53
5.3.1.	Introduction.....	53
5.3.2.	Example.....	53
5.3.3.	Building a PHMM.....	55
5.4.	Oligonucleotide-frequency-models (OFM).....	57
5.4.1.	Introduction.....	57
5.4.2.	Example.....	58
5.4.3.	Adding Pseudocounts.....	58
5.4.4.	Building an OFM.....	60
5.5.	An RNA binding model based on binding energy (<i>RNAbind</i>)	61
5.5.1.	Introduction.....	61
5.5.2.	Example.....	61
5.5.3.	Building a <i>RNAbind</i> model.....	63
5.6.	Scoring the relative position of a signal: <i>DistM</i>	64
5.6.1.	Introduction.....	64
5.6.2.	Building a <i>DistM</i> model.....	65
6.	Combining models	66
6.1.	Hybrid Models: Adding or subtracting scores of several models	66
6.1.1.	Building a hybrid model	68
6.2.	Combining models using decision trees	69
6.2.1.	Introduction.....	69
6.2.2.	Construction of a decision tree	70
6.2.3.	Optimisation of thresholds	73
6.3.	Combining models using neural networks	76
6.3.1.	Introduction.....	76
6.3.2.	Training of backpropagation networks.....	78
6.3.3.	Over-learning.....	78
6.3.4.	Example of a neural network for the human acceptor splice site	79
6.4.	Over-learning	87
7.	<i>FastAFormat</i> - formatting sequences and much more.....	88
7.1.	Functions	88
8.	Distribution and download of models from the <i>Seqool</i> website	91
9.	References.....	93
10.	Appendix	94
10.1.	File formats used in <i>Seqool</i>	94
10.2.	Performance of example models	96

Seqool - A Sequence Analysis Tool

1. Overview

Seqool is a sequence analysis tool designed primarily for searching biological signals in nucleic acid sequences, providing several methods for pattern analysis and the most common basic sequence statistics. It comprises methods for text search (IUPAC nucleic acid codes are supported, such as 'y' for pyrimidines), and models for searching biological signals. The implemented models include profiles/position specific score matrices, profile hidden Markov models, maximum dependence decomposition models, and oligonucleotide frequency models. Models can be combined in several ways (e.g. by decision trees or backpropagation networks), allowing a combination of rather site specific models such as profiles and nucleotide composition in the proximity of a signal. Additional features include the calculation of sequence composition (GC, codon usage, nucleotide and oligonucleotide frequencies) and a manipulation and extraction tool for sequences and text. The major features of **Seqool** include:

Basic sequence analysis:

- Nucleotide composition, oligo-nucleotide composition
- GC content, codon usage, codon preference
- Over- or under-represented oligo-nucleotides
- Calculation within windows of a given size or for whole sequences, for single sequences or several sequences together

Signal search:

- Exact text search, text search using IUPAC codes (e.g. "y" for pyrimidines), search of repeats, stop and start codons, restriction sites
- Profiles (weight matrices/position specific score matrices)
- Profile hidden Markov models
- Maximum dependence decomposition
- Oligo-nucleotide frequency models / models for sequence composition (e.g. GC, codon usage, codon preference, frequencies of nucleotides or oligo-nucleotides)
- Search for RNA binding motifs (based on binding energy)

Combination of models:

- Decision trees
- Neural networks (Backpropagation networks)
- Model combinations by addition or subtraction of scores (Hybrid models)
- Models for the distance between signals

File format support:

- Support of the most common sequence file formats, such as FastA, GenBank, GCG, EMBL, and plain sequences (raw).
- A comprehensive sequence and text formation and extraction tool (FastAFormat) which allows the extraction of sequences from virtually any file format.

The **Seqool** program package consists of two main applications, **Seqool** for the analysis of single sequences and **SeqoolM** for multiple sequences. A variety of independent applications allow the construction and combination of individual models, which can subsequently be applied for searching in the main applications:

Main applications:

- **Seqool** Analysis of single sequences
- **SeqoolM** Analysis of multiple sequences

For building single signal search models:

- Profile Profiles (weight matrices/position specific score matrices)
- PHMM Profile hidden Markov models
- MDD A combination of several profiles using maximum dependence decomposition
- OFM A model assessing the frequency of oligonucleotides, GC, or codon usage
- RNAbind A simple model calculating the binding energy to a given target oligonucleotide

For combining and the classification single models:

- DistM Calculation of the position of the best hit found by a specific model
- HyM A simple combination of single models, summing up scores of single models
- BPNet A neural (backpropagation) network for the classification of signals
- DecisionTree A decision tree used for the classification of signals

Sequence manipulation tool:

- FastAFormat Extraction, manipulation, and formatting of sequence or text files
Functions include: Extraction of sequences from text files, conversion of sequences to FastA format, filtering of sequences according to their length, creation of homologous or reverse strands, extraction of subsequences / truncation of sequences, extraction of columns, truncation of lines, extraction of substrings (optional conditions provided), filtering of lines (optional conditions provided), replacement of text / case conversion, extraction of random lines / sequences, etc.

2. Installation

Download and installation

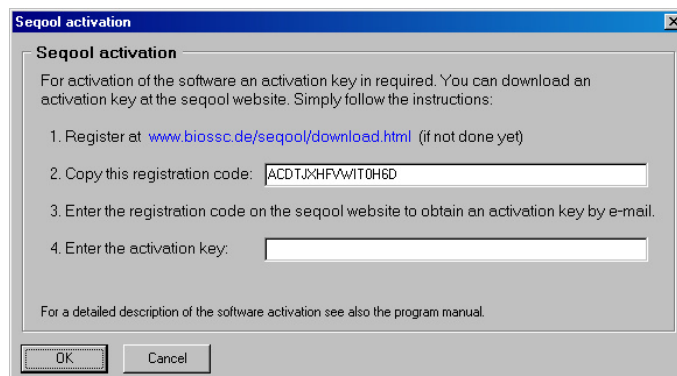
Seqool is provided as a Win32 application running on Windows 2000 and Windows XP systems. The following steps describe the installation of the software. For downloading the **Seqool** program package it is necessary to register at the **Seqool** website. Registration offers several advantages to users: Besides the free download of the program package, users can download pattern recognition models and use them with **Seqool** on their local computers. Users may also upload their own models, if they are of scientific significance.

1. Access the **Seqool** download area at <http://www.biossc.de/seqool/download.html> (registration/login required).
2. Download the file "Seqool_install.exe".
3. Install **Seqool** on your computer by executing the file "Seqool_install.exe". Follow all instructions during installation.

Activation of the software

When one of the programs **Seqool** or **SeqoolM** is started the first time after installation, an activation key is needed for activation of the software. The activation key can be obtained from the **Seqool** download area at <http://www.biossc.de/seqool/download.html>.

1. Start **Seqool** or **SeqoolM**. The following screen shows up:



2. Copy the registration code to the clipboard.
3. Access the **Seqool** download area at <http://www.biossc.de/seqool/download.html> and select "Activation key".
4. When you enter the registration code and press "Get activation key" you will receive an e-mail containing the activation key.
5. Copy the entire activation key and paste it the screen which shows up when **Seqool** or **SeqoolM** is started the first time after installation (see above), and Press OK.

3. Analysis of a single sequences with Seqool

Seqool and **SeqoolM** are the main application of the **Seqool** package. These programs provide a number of sequence analyses and text search functions, and allow to search for biological patterns with user build models. Most functions are identical in **Seqool** and **SeqoolM** (though **SeqoolM** provides some additional functions specifically for multiple sequences). However, the text output and display is optimized for multiple sequences in **SeqoolM** and for single sequences in **Seqool**.

3.1. Basic orientation in Seqool

Seqool provides three main program pages, one for viewing or editing the source file (*Source*), one for the graphic output (*Graphics*) and one for the text output (*Text output*). A panel for displaying hits of text or model searches (e.g. binding sites of restriction enzymes) is located at the top of the graphics page (see Figure 1). Below that panel, additional panels display results of sequence statistics or any other analysis (e.g. graphics for codon usage, stop codon search, etc.). When moving the mouse over one of these graphics, the position of the cursor within the sequence is displayed at the top of the graphics sheet, as well as an excerpt of the sequence at that position. The exact position of the cursor is indicated by a red mark (see Figure 2).

Graphics can be copied to the clipboard or saved using the context menu. This menu appears after pressing the right mouse button when the mouse cursor is located above the respective graph.

The text output page shows results in text-form, if text output was selected in the respective analysis (Figure 3). For analyses for which a graphical output is not meaningful (e.g. for calculation of the codon usage of a whole sequence) text output will always be provided, even if it was not explicitly selected. Tables in the text output sheet are separated by tab-stops in order to allow to transfer them easily to other computer applications, such as spreadsheet calculation programs, by copy and paste. The entire text output can be deleted by pressing the *Clear* button, or saved to a text file by pressing the button *Save as*. *Word wrap* adds line brakes to long lines.

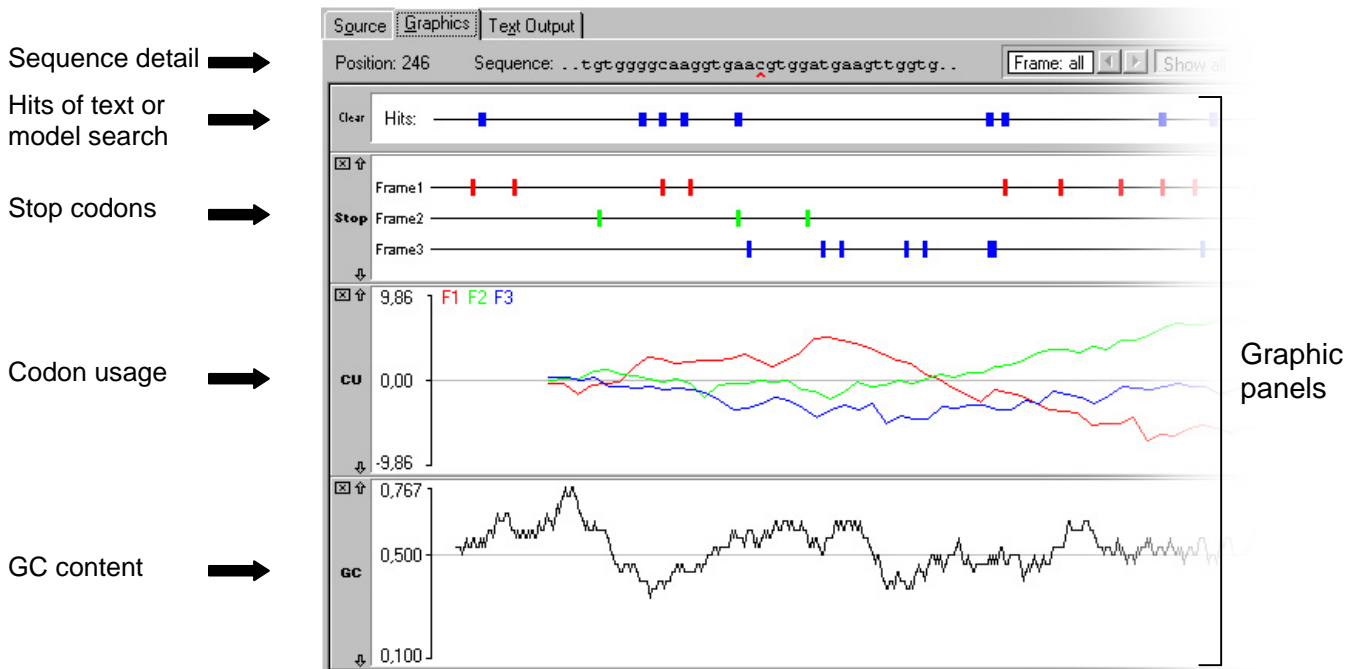


Figure 1. The *Graphics* page displays hits of text or model searches, and graphics of several sequence statistics, such as codon usage, GC content, and others.

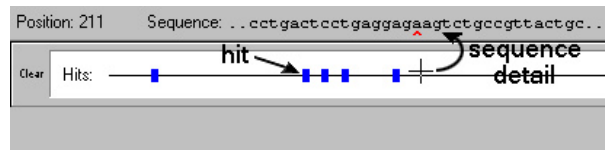


Figure 2. Hits of text or model searches are displayed in a panel at the top of the *Graphics* page. When moving the mouse over this panel or any other box, an excerpt of the sequence around the position of the mouse cursor is displayed at the top of the *Graphics* page.

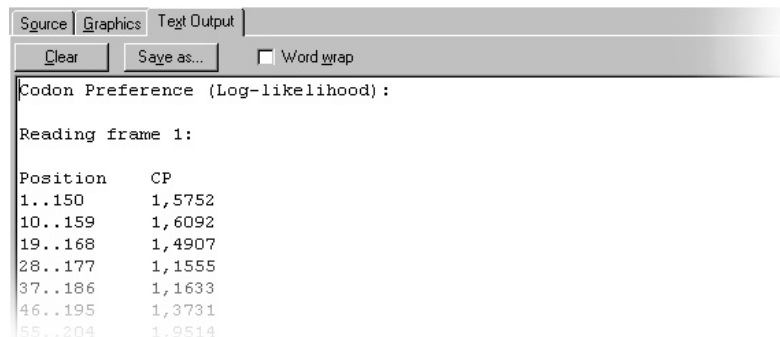


Figure 3. The *Text output* page provides detailed information about the results of an analysis.

3.2. Opening files

Seqool reads FastA, GenBank, EMBL, CGC, and plain sequence files. When a file is opened the sequence format is automatically determined. The raw sequence is displayed in a small window and the user is prompted to confirm the file format (or optionally change the sequence format, Figure 4). Once a file is opened, information about the sequence is displayed at the top of the window, just below the tool buttons (including the sequence length, the format, the molecule, and the ID or locus information; see Figure 5). This information can be hidden by deselecting the menu option *View>Show sequence information*.

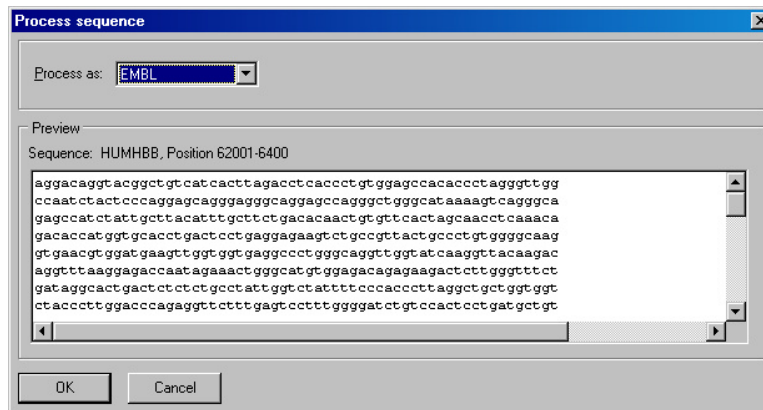


Figure 4. The sequence dialog for confirming or changing the file format of files opened with **Seqool**.

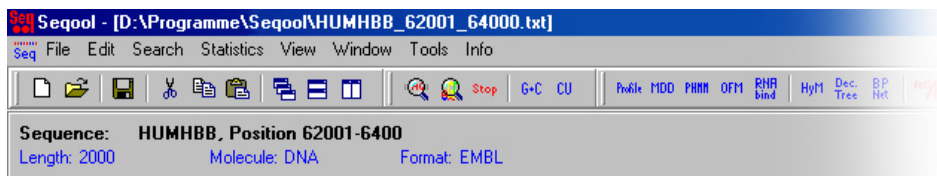


Figure 5. Information about the name, length, molecule, and format of a sequence is shown at the top of the window of **Seqool**.

3.3. Sequence composition

3.3.1. Base composition, GC content, nucleotide/dinucleotide frequencies

For analysing base composition, nucleotide or dinucleotide composition press the button **G+C** or select *Statistics|Base composition* in the menu. A new window with several options appears (Figure 6). In this window select first which statistic to calculate, i.e. GC content, nucleotide frequencies, or dinucleotide frequencies. Then select if the respective statistic shall be calculated for a sliding “window” or for the whole sequence. If a sliding window is used, the statistic is calculated for each subsequence of a specific length, e.g. 30 nt (the “window”). After each calculation, the window is moved to the right by a given amount, e.g. 1 nt (the step), and the statistic is calculated again for the new window. This procedure is repeated until the window has slid over the whole sequence. When using a sliding window specify the width of the window and the amount the window is moved after each calculation.

Optionally select the option *Text output* for more detailed information, e.g. specific values for each position of the sequence.

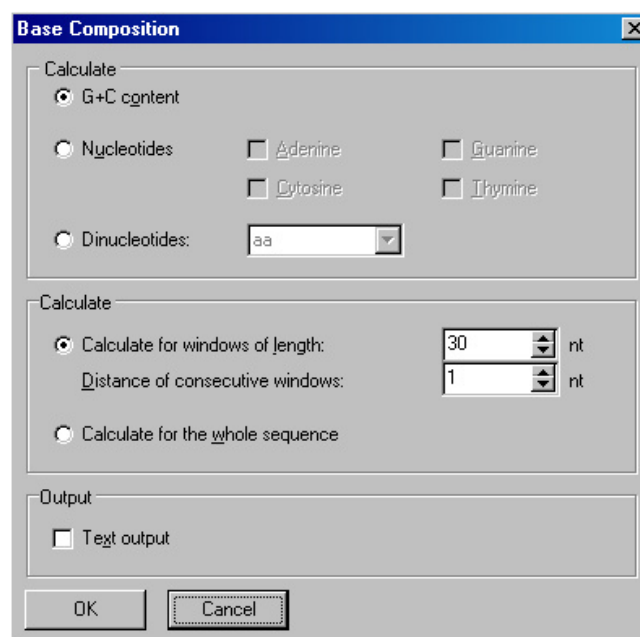


Figure 6. The window for calculating base composition, GC content, and dinucleotides frequencies.

3.3.2. Codon usage, codon preference

Codon usage measures reflect the probability that a sequence is coding. They are calculated using codon usage tables, which are available for many organisms. In **Seqool**, a codon usage table for humans is used by default after installation, however tables for other species can be applied too (see below). For analysing codon usage or codon preference press the button **CU** or select *Statistics|Codon usage* in the menu. The subsequently appearing window shows several options (Figure 7). First, choose if codon usage or codon preference shall be calculated. Codon usage reflects the probability that a sequence is coding in a given reading frame. Codon preference takes only the uneven use of synonymous codons into account (Gribbskov et al. 1984). Codon usage and codon preference are usually calculated for reading frames, thus the option *Calculate for frames* should normally be activated (otherwise values are calculated regardless of frames). Select if the respective statistic shall be calculated for a moving “window” or for the whole sequence. If a window is used, the statistic will be calculated for each subsequences within a sliding window of a given length, e.g. 50 codons (the “window”). After each calculation the window is moved to the right by a given amount, e.g. 3 codons, and the statistic is calculated again for the new window. This procedure is repeated until the window has slid over the entire sequence. When using a sliding window specify the width of the window and the amount the window is moved after each calculation. Optionally select the option *Text output* for more detailed information, e.g. specific values for each position of the sequence.

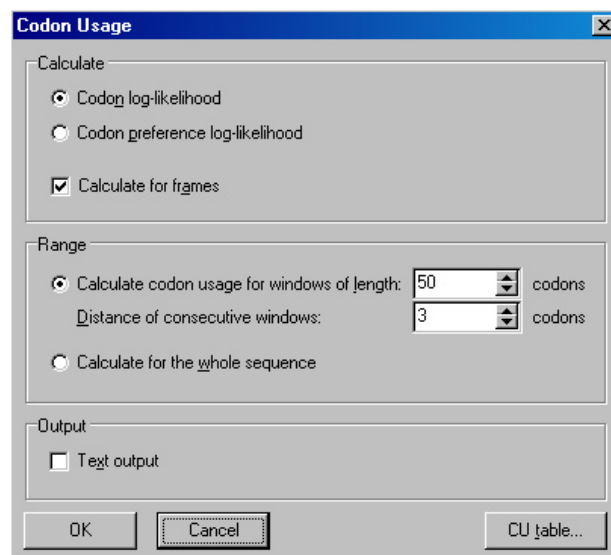


Figure 7. Options for calculating codon usage and codon preference.

For changing the currently used codon usage table press the button *CU table*, which opens a new window displaying the current codon usage table. This table lists all codons, the frequency

of each codon (per 1000), the frequency relative to all synonymous codons, and the number of synonymous codons. Codon usage tables for other species can be loaded by pressing *Open* (make sure to provide tables with an identical format as the default table, separating column with spaces!). A codon usage table can be used as default by pressing *Set as default*, so that it is automatically loaded at the next program start. Alternatively, a default codon usage table can be specified in the Configure-dialog (see menu item *File|Configure ...*).

Codon usage

Codon usage is calculated as the log-likelihood for a sequence to be coding, based a codon usage table which contains the frequencies of codons in coding regions of a species. Codon usage is calculated as follows:

$$CU = \log \frac{\prod_{i=1}^n P_{coding}(C_i)}{\prod_{i=1}^n P_{not\ coding}(C_i)}$$

$P_{coding}(C_i)$ is the probability (or frequency) for an observed codon C_i to be coding, $P_{not\ coding}(C_i)$ is the probability of the observed codon not to be coding (i.e. $1/64 = 0.0156$ assuming a random codon usage in non-coding sequences), n is the number of codons in the sequence (in a given reading frame. For example, the codon usage for the first reading frame of the sequence AGGTAT is calculated as follows (assuming a human codon usage):

$$CU = \log \frac{P_{coding}(agg) \cdot P_{coding}(tat)}{P_{not\ coding}(agg) \cdot P_{not\ coding}(tat)} = \log \frac{0.01209 \cdot 0.01180}{0.0156 \cdot 0.0156} = -0.232$$

Codon preference is the log-likelihood for a sequence to be coding based on the uneven use of synonymous codons, i.e. different codons coding for one and the same amino acid. This statistic is calculated as follows:

$$CP = \log \frac{\prod_{i=1}^n \frac{P_{coding}(C_i)}{1 - P_{coding}(C_i)}}{\prod_{i=1}^n \frac{1}{m_j}}$$

$P_{coding}(C_i)$ is the probability for an observed codon C_i to be used in favour of the its synonymous codons (used with the cumulative probability $1 - P_{coding}(C_i)$) when the sequence is coding. m_j is the number of synonymous codons for a given codon j , n is the number of codons in the sequence.


3.4. Searching for patterns

3.4.1. Text search

3.4.1.1 Introduction

Text search provides a simple way to search for patterns. Text search in **Seqool** is facilitated by the use of IUPAC nucleic acid codes, such as 'y' for pyrimidines, or 's' for strong bases (G or C). For example, searching for the string 'yyyy' will identify most polypyrimidines. **Seqool** implements two methods for text search. One method uses a suffix tree, which also allows to identify repeated elements, the other methods uses the standard Knuth-Morris-Pratt search algorithm. Both methods perform equally well in most cases. However, when searching for many signals, the suffix tree method may be more effective.

3.4.1.2 Basic text search

Open the text search dialog by pressing the button  or by selecting *Search|Text search* in the menu. The search window provides a number of option (Figure 8). The easiest way to search for a subsequence is to enter the search string, i.e. the subsequence, in the left column of the table (optionally a description can be provided in the right column for each search string; this description will appear in the graphical display of the hits, see Figure 9). If IUPAC codes are used, make sure to activate the option *Use IUPAC nucleic acid codes*. A list of all IUPAC nucleic acid codes can be displayed by pressing on the blue arrows behind the check box *Use IUPAC nucleic acid codes*. Start the search by pressing the *Search* button. Subsequently hits are displayed in the hit panel at the top of the graphics page.

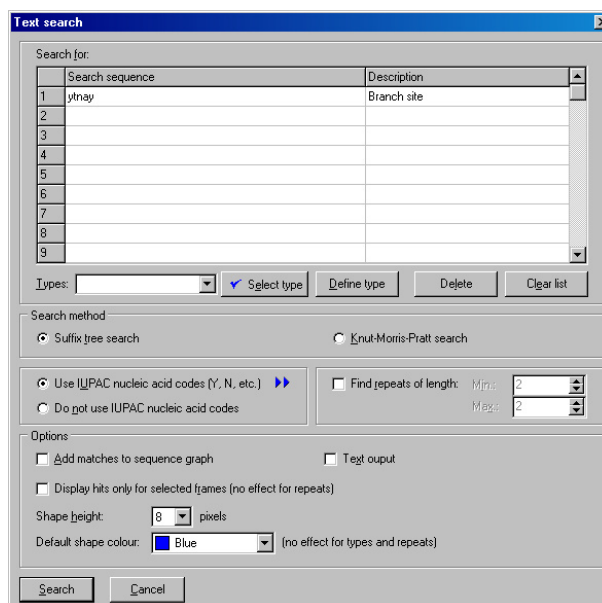


Figure 8. The text search dialog of **Seqool** supports IUPAC nucleic acid codes and a search for repeats.

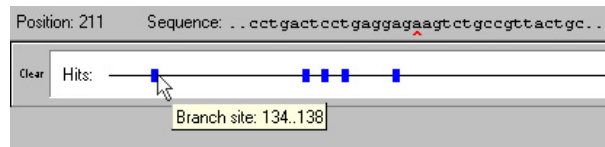



Figure 9. Hits are displayed in the hits panel at the top of the graphics page. When moving the mouse over a hit, a tooltip shows displays additional information of each hit (the description of the search pattern and the position of the hit).

3.4.1.3 Types – Frequently used / alternative search-patterns

Introduction

Frequently, it will be necessary to search for the same patterns routinely. For such cases search *types* can be created. Types can be selected rapidly in the text search window and they may contain more than one search string. For example, the type 'Stop codon' allows to search for all stop codons and includes all three stop codons as search strings, i.e. TAA, TGA, and TAG. Additionally, a specific colour can be assigned to each type for displaying hits (e.g. the colour red for the type 'Stop codon').

Searching with types

For searching with a type open the text search window by pressing the button  or by selecting *Search|Text search* in the menu. Then select a type in the *Types* box on the left below the main search table. Several types are already defined after installation of **Seqool**. Select for example the type 'EcoRI' and press the button *Select type*. Subsequently a new entry is automatically added to the search table: #ecori, with the description 'EcoRI'. Pressing the *Search* button will now search for all restriction sites of the enzyme EcoRI.

Creating new types

Search types are created by pressing the button *Define type*, which opens a new window (Figure 10). The name of a new type, e.g. 'Stop codon', is entered at the top of this window (in the box *Type name*). The search patterns are entered in the main table (for the three stop codons the search patterns are 'taa', 'tag', and 'tga'). Optionally a colour can be selected for displaying hits. For example, select the colour red in the *Type colour* box in order to display stop codons red.

A final option determines if hits of the current type will be displayed only if they occur in a given reading frame (which is appropriate for stop codons) or if they shall always be displayed (as in the case of restriction enzymes), regardless of the reading frame (if hits are displayed for the

respective reading frame only then hits can be displayed for each frame separately, see option *Frame* at the top of the *Graphics* result sheet). Finally, a type is created by pressing the *Add type* button (other types can be deleted by selecting a respective type and pressing the *Delete type* button). **It is important to note that new types are not saved by default. For making types accessible at the next program start types have to be saved in a type file.** Type files include a collection of types. To save changes in a type file after the creation of a new type press *Save type file*. For opening a type file press *Open type file*. When starting **Seqool** a type file is loaded by default. This default file can be selected in the menu *File|Configure*).

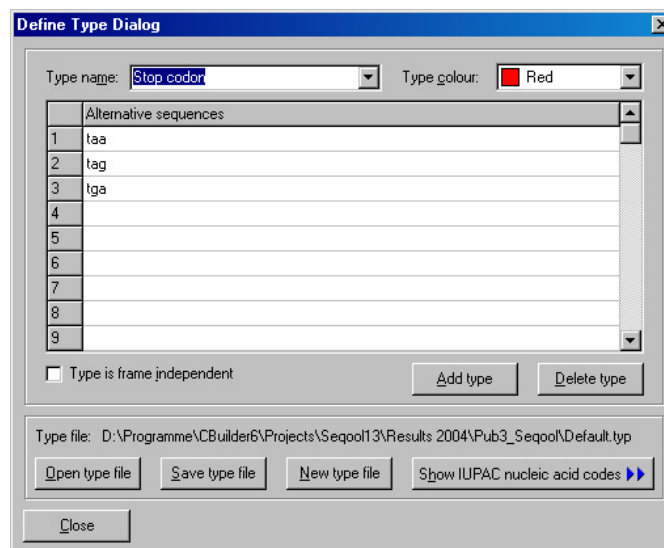


Figure 10. The dialog for defining search types. Search types are used for searching routinely. Search types may include several search patterns, as in this example for stop codons.

3.4.1.4 Searching for repeats

Before searching for repeated subsequences make sure to apply suffix tree search. Activate the option *Find repeats of length* and specify the minimum and maximum length of the repeats to find. Finally press the *Search* button. Repeats are always shown in grey colour in the hit panel of the graphics page. The exact sequence of repeats is shown as a tool tip when the mouse cursor is moved over the respective shape in the hit panel.

3.4.1.5 Additional options

Several options in the text search window allow to customize the graphical output and the text output:

- Add matches to sequence graph - Usually previous hits displayed in the hit box are erased before displaying new results searching. Activation of this option prevents previous hits to be erased.
- Text output - Hit positions are also printed in the *Text output* page.
- Display hits only for selected frames - Hits can be showed only for a specific frame, which is selected at the top of the *Graphics* page. This option is useful e.g. for stop codons, which are frame dependent.
- Shape height - Define the height of the shape for depicting a hit in pixels.
- Default shape colour - Define the colour of the shape for depicting a hit. This option does not apply to search types and repeats.

3.4.2. Stop codon search

Seqool provides a convenient additional option for displaying stop codons in all three reading frames by pressing the button **Stop** or by selecting *Search|Search for stop codons* in the menu. In the following dialog (Figure 11) it can be specified if the stop codons of any frame are displayed in the same colour (red), or if different colours are used, which correspond to those colours used in codon usage graphs (this option facilitates an easier identification of open reading frames or exons). Additionally the height of graphically displayed stop codons can be defined. For an example of the stop codon graphic see Figure 12.



Figure 11. The dialog for stop codon search.

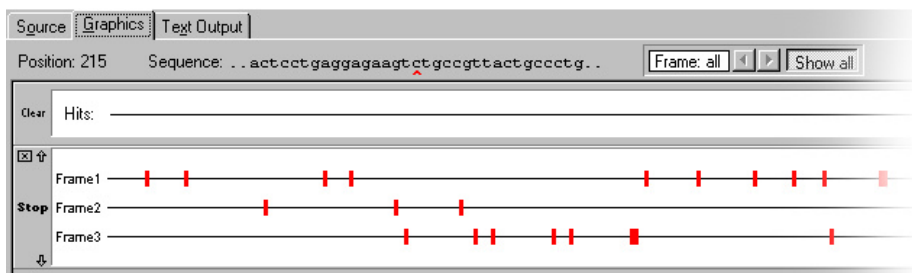


Figure 12. Example of the graphical display of stop codons. Stop codons are displayed for each reading frame.

3.4.3. Searching with pattern recognition models



3.4.3.1 Introduction

The **Seqool** program package provides separate modules for creating a variety of pattern recognition models, such as position specific score matrices (weight matrices), profile hidden Markov models, and models for combining models, such as decision trees or neural networks. Modules for building pattern recognition are started by selecting the respective component in the *Tools* menu or by using the respective tool button (Figure 13). All models can be used for searching in **Seqool** and **SeqoolM**.



Figure 13. Tool buttons for starting program components for signal-recognition models.

3.4.3.2 Basic search

Pressing the button  (or selecting *Search|Model search* in the menu) opens the model search dialog (Figure 14). When searching with a specific model, only models contained in the default model directory are listed. This default directory can be changed permanently in the main menu (menu *File|Configure...*) or only temporarily (until exiting the program) by pressing the *Model directory* button. If a new model was saved in the model directory while the model search dialog was still open it may not be listed in the list of models. In this case press  to actualize the list of available models.

For selecting a model, choose one of the models listed in the *Select model* box and press the button *Add model*. Additional information about the model is displayed below the *Select model* box, such as the model type, the models name, the description, and the file name (see Figure 14). Press *Search* for starting the model search.

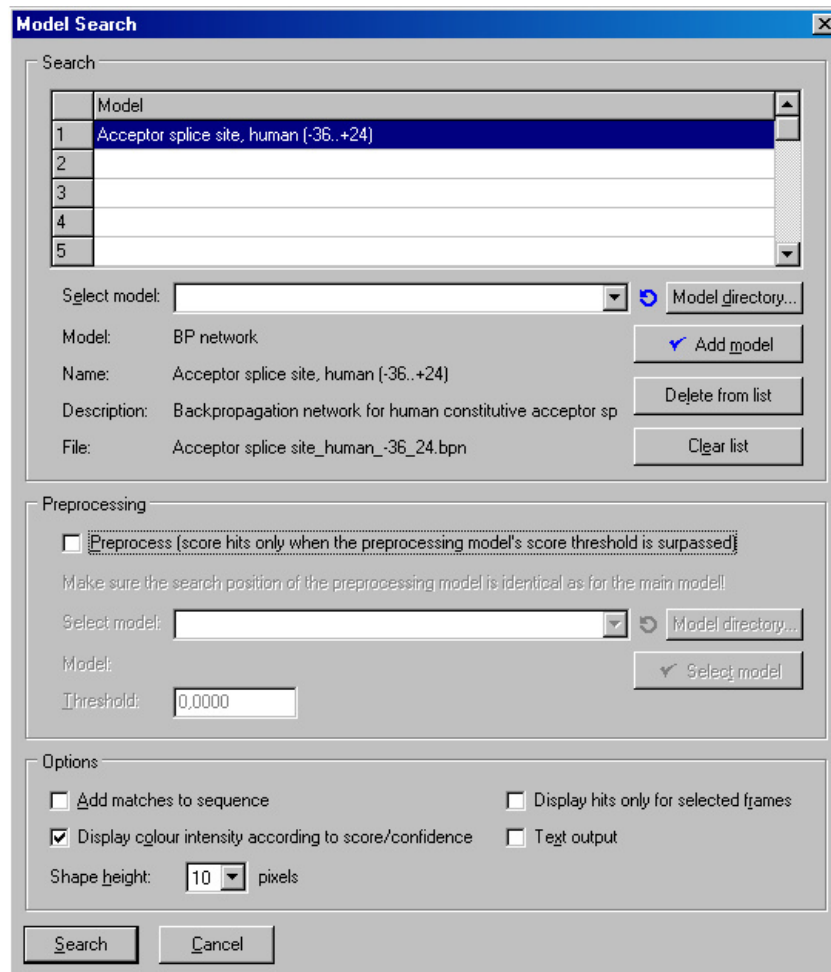


Figure 14. The model search dialog. Pre-processing models can be added before search with the main model.

3.4.3.3 Adding a pre-processing model

In some cases it might be useful to add a pre-processing model before searching with the main model. For example, when searching for acceptor splice sites, it will be faster to search for all AG dinucleotides first and then to apply a more complex model only to those sites, where an AG dinucleotide was found. This may significantly reduce the overall time for searching. A pre-processing model is added by activating the check box *Preprocess* and by selecting a model in the respective *Select model* box. Confirm a selected model by pressing the button *Select model*. **Make sure that the search position of the pre-processing must match the search position of the main model.** For example, when the main model analyses a region between –10 and +10 nt relative to the putative signal, then the pre-processing model must match this region, i.e. the pre-processing model must also start at the position –10 nt (but it does not necessarily end at the position +10 nt). *Hybrid models* (see chapter 6.1) can be used to adjust the range of a pre-processing model. Although the component *Hybrid models* is intended for

combining several models, it provides an easy method for shifting the position of a model (see figure Figure 15). For details refer to chapter 6.1, which provides a detailed description of *Hybrid models*.

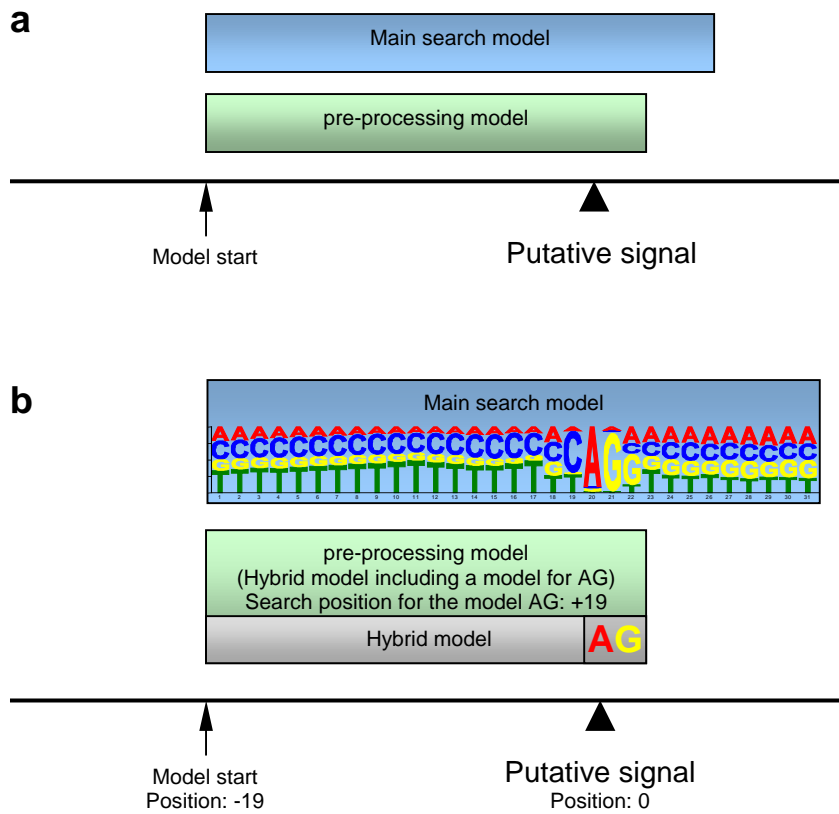



Figure 15. a) A pre-processing model must match the range of the main model. b) The component *Hybrid models* allows to relocate a pre-processing model so that it matches the range of the main model. In this example the pre-processing recognizes the AG dinucleotides. In order to match the search position of this model to the search position of the main model a hybrid model (including the model for AG) is used to shift the pre-processing model to the left back to the correct search position.

When using a pre-processing model, the score threshold may be changed by the user in order to increase or decrease the specificity. The default threshold shown when selecting a model corresponds to the original value given in a model.

As for the main model list before, the model directory can be changed temporarily by pressing the *Model directory* button and the list of models can also be actualised by pressing .

3.4.3.4 Additional options

Several options allow to customize the graphical of text based output:

- | | |
|--|---|
| Add matches to sequence | - Usually previous hits displayed in the hit box are erased before displaying new results searching. Activation of this option prevents previous hits to be erased. |
| Display colour intensity according to score/confidence | - The colour of graphically displayed hits is modified depending on the score of the hit (or confidence in the case of neural networks). A dark colour indicates a high score, while a light colour indicates a low score |
| Display hits only for selected frames | - Hits can be showed only for a specific frame, which can be chosen at the top of the graphics result sheet. This option is useful e.g. for stop codons, which are frame dependent. |
| Text output | - Hit positions are also printed in the text output sheet. |
| Shape height | - Define the height of the shape for depicting a hit in pixels. |

3.5. Additional functions

3.5.1. Exporting sequences

Sequences can be converted to the most common sequence formats using the menu option *File|Export sequence...*. Available sequence formats include FastA, GenBank, EMBL, CGC, and plain sequence files (raw sequence files). Select the desired format in the *Sequence format* box and press *OK* to save.

3.5.2. Creating subsequences

For large sequence the Sequence display in **Seqool** is compressed in order to allow to display the whole sequence within one panel. It may therefore be useful to copy a subsequence, e.g. the region of a putative signal, in a new window in order to study it in more detail. Subsequences are selected by moving the mouse across the region of the sequence which shall be copied (in the hits panel at the top of the graphics page) while holding down the left mouse button. A blue line below the hits-box indicates the selection. After releasing the mouse button a window opens which allows to adjust the selection and to copy the respective subsequence into a new window (the latter option is also available in the menu: *Edit|Copy selection into new file*).

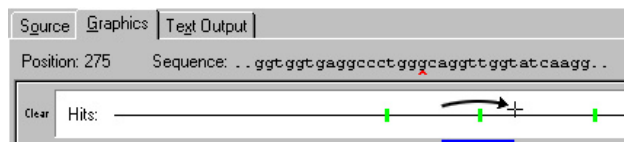





Figure 16. A subsequence can be marked (blue line) by moving the mouse over a part of the sequence while holding down the left mouse button.

3.6. Customizing Seqool

3.6.1. Graphical output

The panels containing the graphical output of an analysis can be deleted by pressing the small  at the top left corner (Figure 17) of the panels. Graphics can be rearranged using the small flash buttons  or  on the top and bottom left side of the graphics (Figure 17). These buttons allow to switch a graphic panel upwards or downwards, respectively.

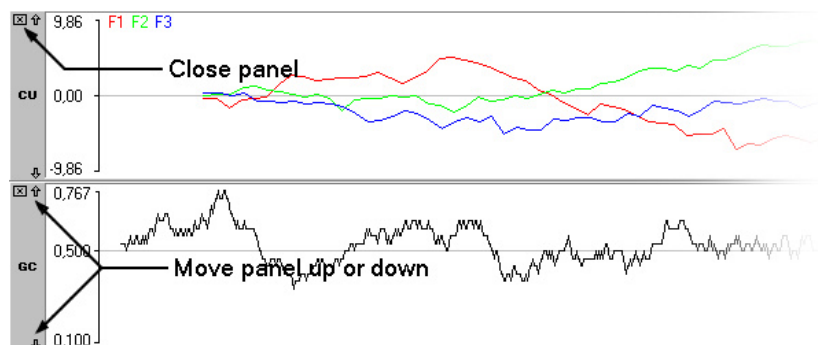


Figure 17. Buttons for closing and moving graphic panels.

3.6.2. Configuring Seqool

Basic settings can be customized using the menu option *File|Configure...*. All options listed in the configure window (Figure 18) are permanent, i.e. the respective setting is loaded by default every time the program is started. The *default type file* refers to the file containing all types which can be used for text search (see chapter Searching with types). The *codon usage file* defines which file is used for the calculation of codon usage of codon preference indices. The model directory determines which folder is used for searching with pattern recognition models (when using a model comprising several submodels please remember that all sub-models must be provided in the same directory as the main model). Finally, it can be selected how many recently opened files are displayed in the file menu.

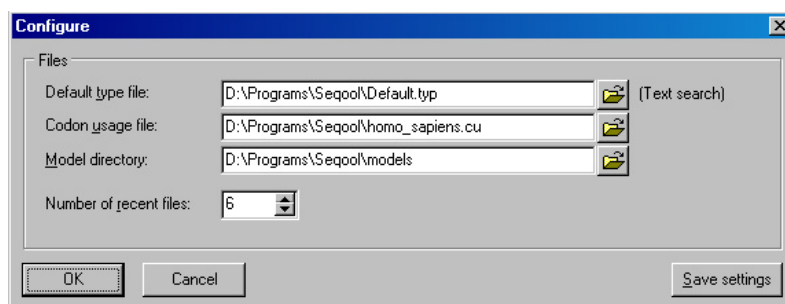


Figure 18. The dialog for customizing basic settings of **Seqool**.

4. Analysis of multiple sequences with SeqoolM

SeqoolM provides almost the same functionality as **Seqool**, such as codon usage, GC content, or text and pattern search. However, the display of results is optimised for the analysis of multiple sequences. Additionally, **SeqoolM** offers analyses which are more relevant for multiple sequences, such as the analysis of over- or under-represented words (i.e. short subsequences), and it provides several filtering options, which can be used for extracting only sequences meeting certain criteria (e.g. a sequences with a GC content of at least 0.5).

4.1. Basic orientation in SeqoolM

SeqoolM provides two main tab sheets, one for the graphical output (*Graphics*) and one for the text output (*Text output*). The graphical results tab sheet consists of two parts: A hit-panel for displaying hits of text or model searches (and also searches for over- or under-represented words, see Figure 19) at the top, and below a box for displaying sequence statistics, e.g. codon usage or GC content. When moving the mouse over one of the graphics, the respective position in the sequence is displayed (see *Position* in Figure 19), as well as the number of sequences having the respective length (since a dataset may contain sequences of different length, the right side (5'-end) of the sequences may be represented by a lower number of sequences than the left (3'-end) side). Similarly, in the hit-box for text or model search grey lines indicate the shortest and the longest sequences (Figure 20).

Graphics can be copied or saved by pressing the right mouse button above the respective graph. The context menu provides options for saving a graph or for copying it to the clipboard.

The text output sheet contains all results in text-form, if text output was selected in the respective analysis (Figure 21). For some analyses text output will always be provided (e.g. for calculation of the codon usage of all sequences). Tables in the text output sheet are separated by tab-stops in order to allow to transfer them easily to other computer applications by copy and paste. All text output can be deleted using the *Clear* button, or saved to a text file using the button *Save as*.

Hits of text or model search (or search for over-/under-represented words)



Graphics for various sequence statistics

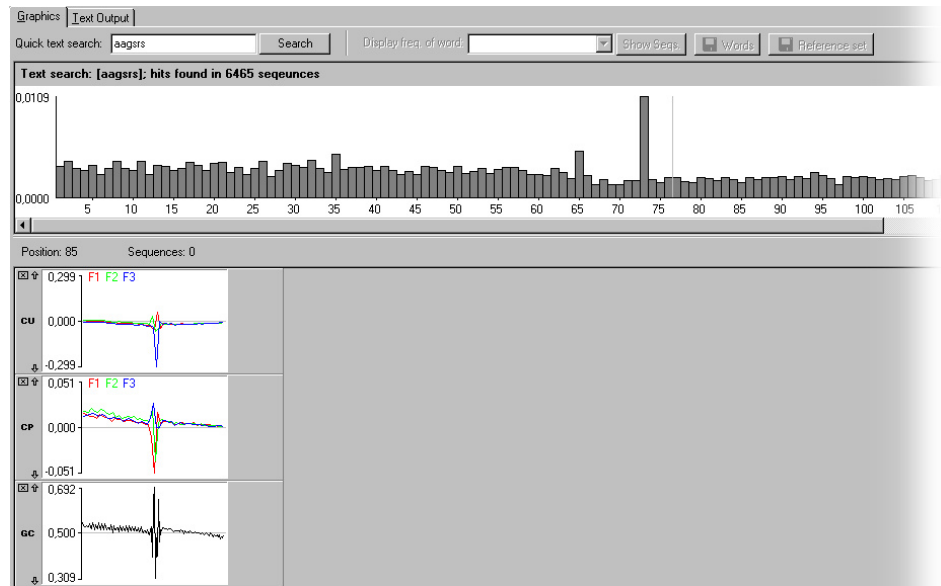


Figure 19. The *Graphics* page displaying hits of text searches, model searches, or searches for over- or under-represented words (top), and graphics for sequence statistics, such as codon usage, GC content, and others (below).

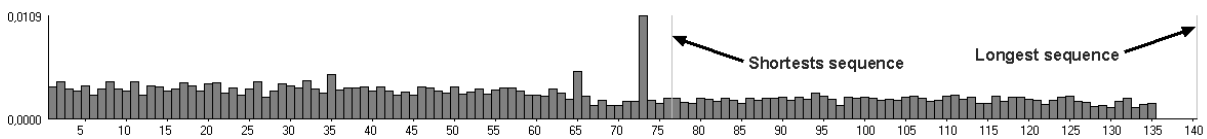


Figure 20. In the hit panel for text or model searches, grey lines indicate the length of the shortest and longest sequences in the data set.

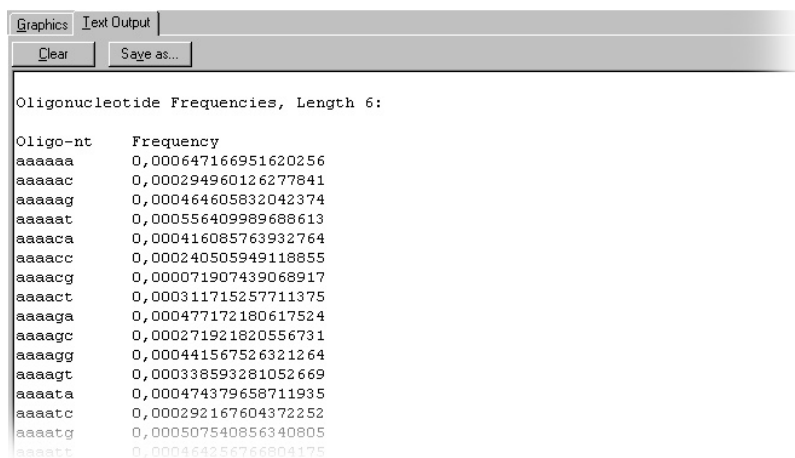


Figure 21. The *Text output* page provides detailed information (optional) about the results of an analysis.

4.2. Opening files

FastA files are opened using the menu item *File|Open* or by pressing the file open button. Before loading a file, FastA files are displayed in a separate window for verification. If files have been correctly read, press *OK*. Figure 22 shows an example of the verification window for the following three sequences:

```
> Sequence A
agggtaggcaggtggctcctctagccccctccaccatcacaggccccaatatgattctcttctcctggcaggtgacgactgacttgcggcagaag
gcagctggagagctgctcccaaaaagactgatccccagcctctcccttcttcttgcagtaagaacctcacatggcgggacatgcaaacct
tccaccatcacaggccccaatatgattctcttctcctggcaggtgacgactgacttgcggcagaagtgcacggagtctcacacgggacactgcag
aaaagactgatccccagcctctcccttcttcttgcagtaagaacctcacatggcgggacatgcaaacctgggtggtacagacctcgattgc
agtaagaacctcacatggcgggac
> Sequence B
gccttagggcagctggagagctgctcccaaaaagactgatccccagcctctcccttcttcttgcagtaagaacctcacatggcgggacat
gcagctggagagctgctcccaaaaagactgatccccagcctctcccttcttcttgcagtaagaacctcacatggcgggacatgcaaacct
gggtggtacagacctcgatcgctaaactggagagctgctcccaaaaagactcctaagaacctcacatggcgggacatgctaagaacctcacatg
gcgggacatgcgatccccagcctctcccttcttcttgcagtaacacatggcgggacatgcaaacctgggtggtacagacctcgactgctgctg
tggtagacacctcgacaaccacgcaccaccactacttgggtgtgagggcagctggagggagctgctcccaaaaagtgtgtgactgatccccag
cctctcccttcttcttgcagtaaga
> Sequence C
gctcgtatctcgtagctgactgactcgcgcgcctctcgtctactatctactatcgtgatgctgctctatctcgtatcgatgcttagcta
ctacgtacgtatagctatcgatcgtgattatcatgctagctagcactctcatgctgctagctccccccccccctgatgctgctgtgatg
acctctctatcgctaataatcgtracgtatgactcagactagatatatcgtcagcttagctacactctcgatcgatcgcgcgctatctcgt
acatgctgagtgtagctctcgatgctgtagctgactgactgatcgatcgtctctccccctatatactcgatctctctccccccca
accggctagctgctacatctctagtggcg
```

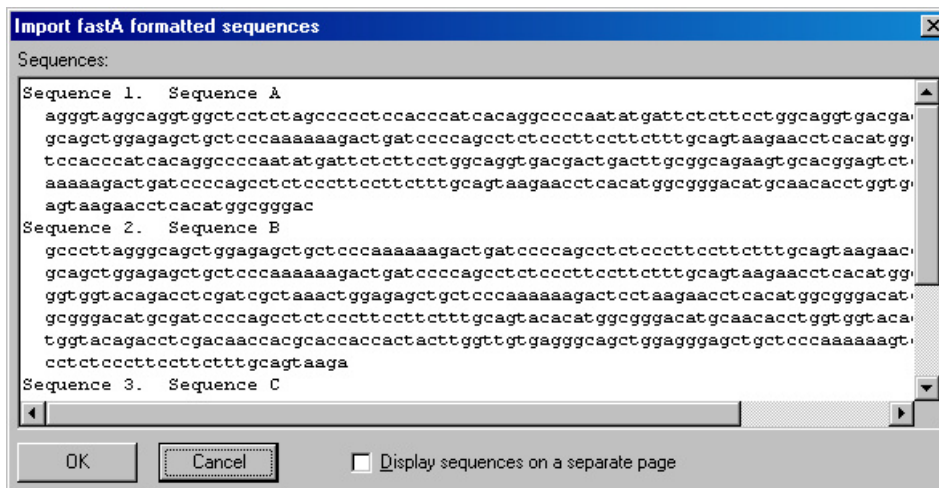


Figure 22. The open dialog for verification of FastA files opened with **SeqoolM**.

In the window for verification of the opened sequences the option *Display sequences on a separate page* allows to show all single sequences on a separate page (see Figure 22). When this option is activated a new page apart from the *Graphics* and the *Text output* pages will show a list of all sequences.

After opening a multiple sequence file information about the sequences is displayed at the top of the window (including the number of sequences in the file, the molecule, and the length of the shortest and the longest sequence; see Figure 23). This information can be hidden by deselecting the menu option *View>Show sequence information*.

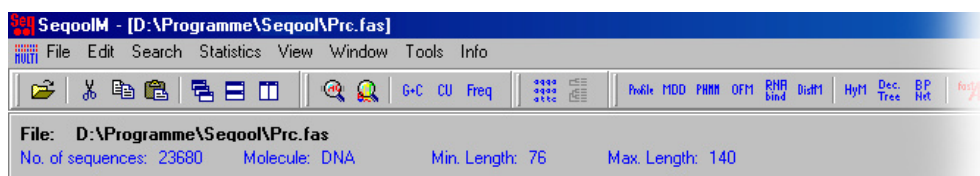


Figure 23. Information about the name, number of sequences, molecule, and length of the shortest and longest sequence is shown at the top of the window of **SeqoolM**.

4.3. Sequence composition

4.3.1. Base composition, GC content, nucleotide / dinucleotide frequencies

For analysing base composition, GC content, nucleotide or dinucleotide composition press the button **G+C** or select *Statistics|Base composition* in the menu. The following window offers various options (Figure 24). First, choose to calculate either GC content, nucleotide frequencies, or dinucleotide frequencies. Select if the respective statistic shall be calculated for a moving “window” or if a single value shall be calculated for a given range of the sequences. If a window is used, the statistic will be calculated for each subsequences within a sliding window of a given length, e.g. 30 nt (the “window”). After each calculation the window is moved to the right by a given amount, e.g. 1 nt, and the statistic is calculated again for the new window. This procedure is repeated until the window has slid over the entire sequence. When using a sliding window specify the width of the window and the amount by which the window is moved after each calculation. Optionally select the option “text output” for more detailed information, e.g. the values for each position of the sequence.

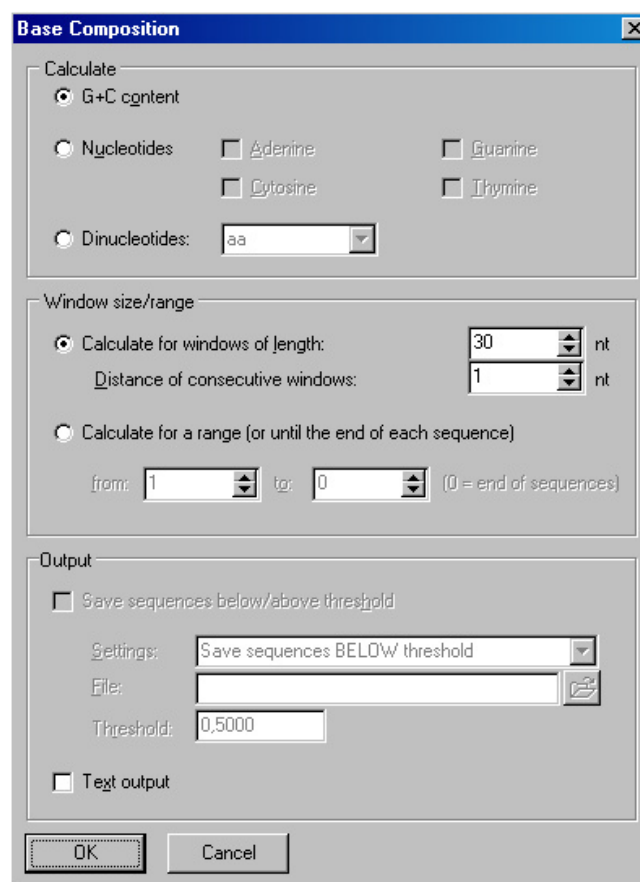


Figure 24. The dialog box with several options for calculating base composition, nucleotide and dinucleotide composition, and GC content.

For calculating a statistic only for a given range specify the range (settings *from* and *to*; entering '0' in the last field will calculate the statistic until the end of each sequence). Those sequences with a value below or above a certain threshold can be saved in a separate file (FastA format). This allows to split a dataset, for example according to GC content (see example in Figure 25).

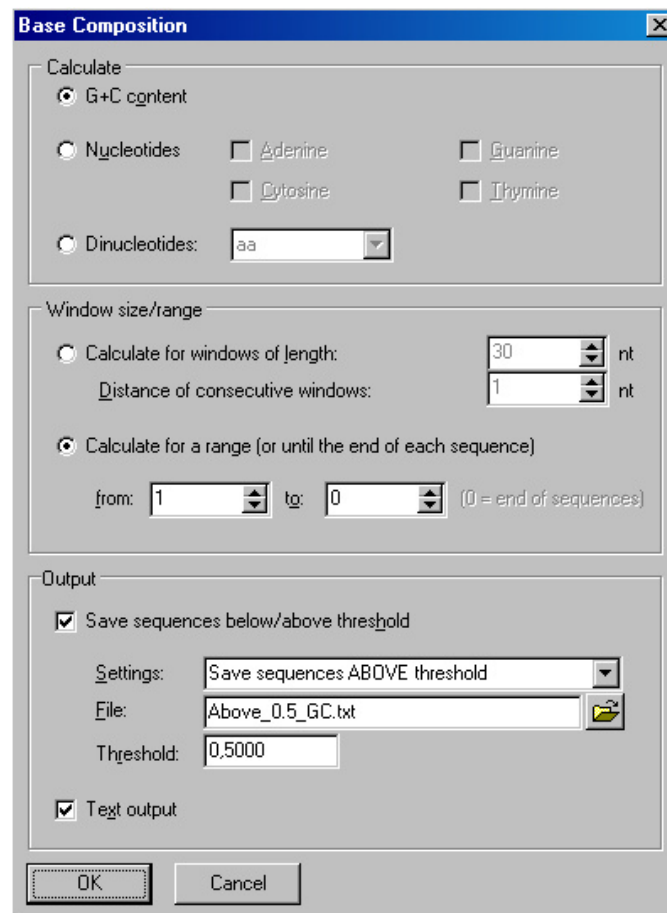


Figure 25. Example for extracting sequences with a GC above 0.5. Sequences are saved in FastA formatted file.

4.3.2. Codon usage, codon preference

Codon usage measures reflect the probability that a sequence is coding. They are calculated using codon usage tables, which are available for many organisms. In **SeqoolM**, a codon usage table for humans is used by default after installation, however, tables for other species can be used as well (see below). For analysing codon usage or codon preference press the **CU** button or select *Statistics|Codon usage* in the menu. A dialog box opens showing several options (Figure 26): First, select either codon usage or codon preference. Codon usage reflects the probability that a sequence is coding in a given reading frame. Codon preference takes only

uneven usage of synonymous codons into account (Gribkov et al. 1984). For calculations of codon usage and codon preference see chapter 3.3.2. Usually, codon usage or preference is calculated for all three reading frames, hence the option *Calculate for frames* should normally be activated. Select if the respective statistic is calculated for a moving “window” or for all sequence. If the a window is used, the statistic is calculated for subsequences of a given length, e.g. 50 codons (the “window”). After that, the window is moved to the right by a given amount, e.g. 1 codon, and the statistic is calculated again for the new window. This procedure is repeated until the window has slid over the whole range. When using a sliding window specify the width of the window and the amount the window is moved after each calculation. Optionally select the option *Text output* for more detailed information, e.g. for specific values for each position of the sequence.

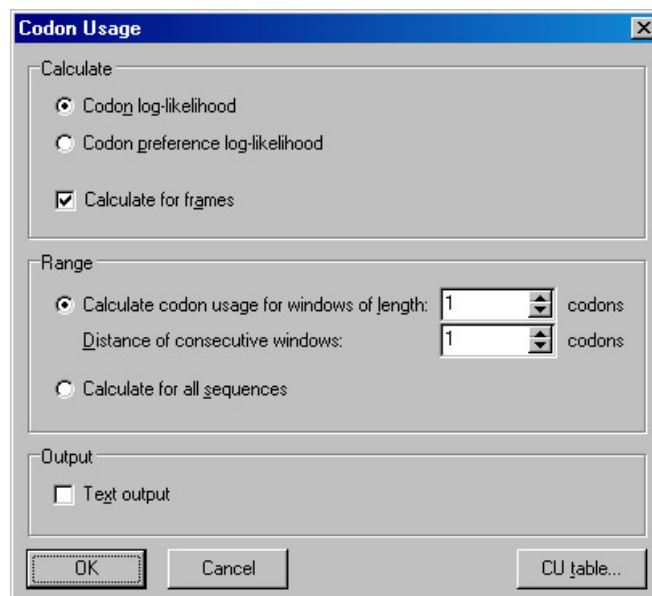


Figure 26. The dialog for calculating codon usage.

The currently used codon usage table can be chosen by pressing the *CU table* button, which shows a new window containing the current codon usage table. This table lists the codons, the frequency of each codon (per 1000), the frequency relative to synonymous codons, and the number of synonymous codons. Other tables can be loaded by pressing *Open* (make sure to provide tables with an identical format!). A codon usage table can be used as default by pressing *Set as default*. The default codon usage file is always loaded at the start of the program. Alternatively, a codon usage table can be set as default in the Configure-dialog (see menu *File|Configure|Settings ...*).

4.3.3. Calculation of oligonucleotide frequencies

For calculating oligonucleotide frequencies press the button **Freq** or select *Statistics|Oligonucleotides* in the menu. In the window which appears (Figure 27) select the length of the oligonucleotides for which frequencies shall be calculated and press *OK*. In some cases (especially for long oligonucleotides and datasets containing only few sequences) some oligonucleotides may not be found, i.e. their frequency in the sample is zero. However, in reality these oligonucleotides may in fact occur, though rarely. Oligonucleotides remain usually undetected when a sequence set contains too few sequences. Frequencies with a value of zero may cause problems when using them for later calculations, e.g. for a recognition model (provoking e.g. division by zero errors). To avoid this problem, frequencies equal to zero can be substituted by a fixed, user defined value (option *For missing oligonucleotides use a frequency of...*). For estimating the expected frequency of an oligonucleotide which was not observed in a dataset, it might help to consider the probability to observe the oligonucleotide by chance (assuming that each nucleotide occurs with the same frequency). For example, the expected frequency for a trinucleotide is $1/4^3 = 1/64 = 0.0156$. For a hexanucleotides the frequency of each hexanucleotide is $1/4^6 = 1/4096 = 0.000244$ (evidently, in a dataset containing 4000 oligonucleotides of length 6 nt it is impossible to find all 4096 hexanucleotides).

Oligonucleotide frequencies can either be shown in the *Text output* page (when the option *Text output* is activated), or they can be saved to a separate file (option *Save frequencies in a file*). This option creates a **Seqool** frequency file (see Appendix 10.1).

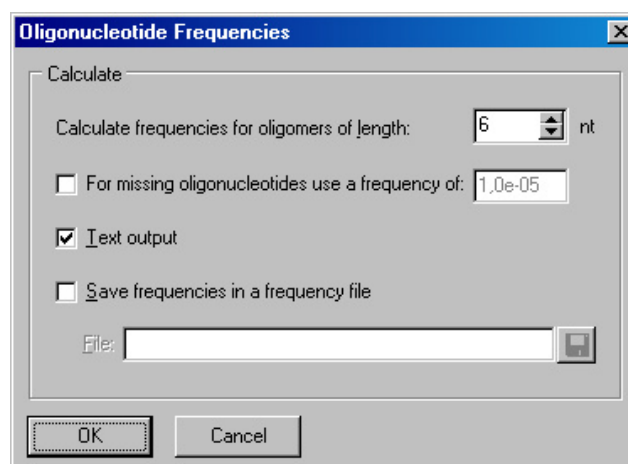


Figure 27. The dialog for the calculation of oligonucleotide frequencies.

4.4. Over- and under-represented words

The frequency of short subsequences of only a few nucleotides, so called *words*, can be compared to the frequency which would be expected given a certain sequence composition. As a simple example consider trinucleotides (words with a length of 3 nt). In non-coding regions with an equal probability of each trinucleotide, a given trinucleotide is expected to occur with a frequency of 1/64. Trinucleotides occurring more frequently than $1/64 = 0.0156$ are over-represented, while others occurring more rarely are under-represented. Over- or under-represented words (in this case the trinucleotides) may indicate e.g. regulatory sequences.

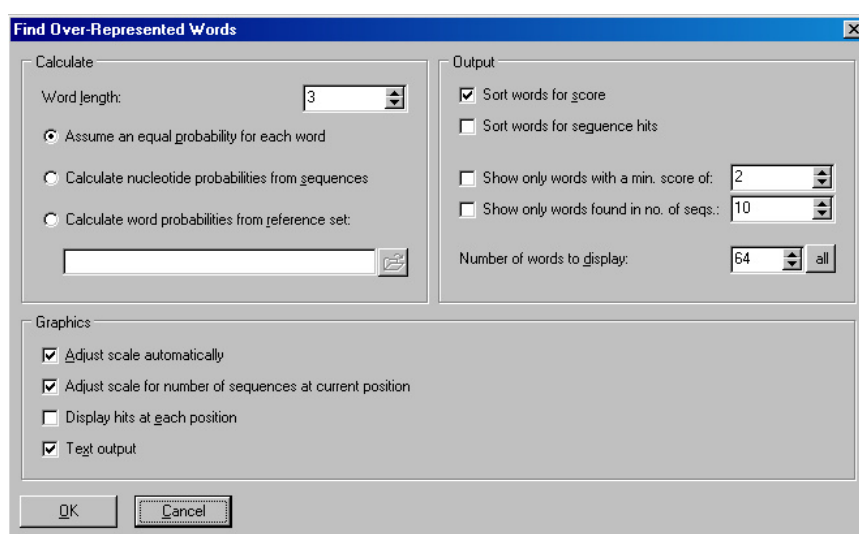



Figure 29. The dialog for calculation of over- and under-represented words (short subsequences).

4.4.1. Calculation of over-/under-represented words

For determining over- or under-represented words press the button  or select *Statistics|Find over- and under-represented words*. The next dialog provides several options (Figure 29): First select the length of the words (short subsequences) which shall be analysed, e.g. select a length of 3 nt for analysing trinucleotides. Then specify how to calculate the expected word frequencies. In the simplest case, all words are expected to occur at random, each with an equal probability (option *Assume an equal probability for each word*, e.g. a probability of 1/64 is expected for trinucleotides). Alternatively, a nucleotide distribution can be calculated from the existing sequences, and the expected occurrence or words can be estimated from this distribution (option *Calculate nucleotide frequencies from sequences*). Since this method provides only a very approximate calculation the option *Calculate word probabilities from reference set* should be used preferentially. This method retrieves the expected probability for each word from a separate file, the *reference set*. Such a reference set contains, for example, a

distribution of trinucleotides in coding DNA or hexamer-frequencies in exons (e.g. for identifying over-represented words, i.e. putative regulatory elements, in exons). See below for a description of how to create reference set files. Further options are described later.

After pressing the button *OK*, scores are calculated for all words and for all positions of the sequences. The frequency of each word at each position is shown in Figure 30. The respective image can be copied or by pressing the right mouse button above the image (follow the option in the appearing context menu). Specific words can be selected in the box *Display freq. of word*. The text output (in the Text output tab-sheet) provides additional information, such as the score of each word, the number of observed and expected hits, etc..

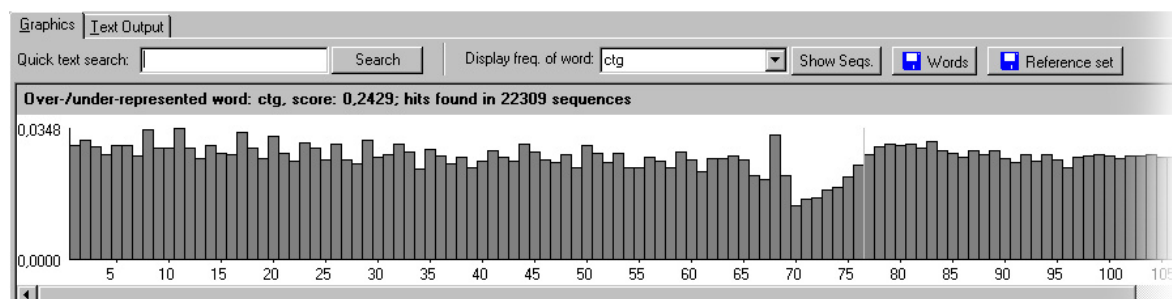


Figure 30. Graphical output of the calculation of over- or under-represented words.



Additional options in the dialog shown in Figure 29 allow to sort or filter certain words. Usually words are sorted for their score (option *Sort words for score*), i.e. over-represented words (with a positive score) appear at the top of the output list and under-represented words appear at the end of that list. Alternatively, words can be sorted according to the number of sequences in which a word was found (option *Sort words for sequence hits*). Words can be filtered according to a certain score threshold (option *Show only words with a min. score of*) or to a minimum number of sequences in which they were found (option *Show only words found in no. of seqs.*). Finally the number of words can be limited to a given number (e.g. show only the ten words with the highest score; option *Number of words to display*).

A few options allow to customize the graphical output and the text-output: *Adjust scale automatically* adjust the scale of the graphic output according to the observed frequencies. If this option is deactivated, the y-axis shows the full range from 0.0 to 1.0. The option *Adjust scale for number of sequences at current position* causes the graphic to display the frequencies of a word relative to the number of sequences in the dataset which cover the position where a word was found. This option is useful if a dataset contains sequences with varying lengths. For

example, consider a dataset contains three sequences of varying length, e.g. 40 nt, 60 nt and 100 nt. If a given word is found at the 50th position in one of these three sequences then the absolute frequency would be 0.333. However, only two sequences provide information about the position 50, hence the relative frequency is 0.500 rather than 0.333.

Further options include the option *Display hits at each position*, which allows to show the number of hits at each position of the sequences in the graphical output, and the option *Text output*, which allows to display a table containing the number of observed and expected hits for each word, the number of sequences in which a word was found, the score, and the number of hits at each position in the sequences in the *Text output* page.

Creating reference set files

Reference sets files are also created by calculation of over-/under-represented words: Press the button  or select *Statistics|Find over- and under-represented words*. In the dialog which opens select the length of the word and choose the option *Assume an equal probability for each word* and press OK for starting the search. When the search is complete, press the button  Reference set at the top of the *Graphics* page to save the reference set (a short message will appear describing the use of a reference set).

4.4.2. Clustering of over-/under-represented words

Over- or under-represented words may include regulatory or otherwise biologically significant signals (e.g. the binding site of a protein). The composition of such elements may be very variable, though some similarities are usually shared between the different sequences corresponding to the signal. The clustering of over-represented sequences provides a method for identifying groups of similar sequences which may correspond to a biological signal. These groups (clusters) of similar sequences can be used for building a recognition model for a signal, e.g. a profiles or weight matrices (see chapter 5.1). The applied clustering algorithm is based on sequences identity as a measure of genetic distance. Clusters are produced using a neighbour joining algorithm.

Identification of over-represented words

As an example, a dataset containing the terminal 70 nt of human introns will be used. For these sequences, over-represented words of length 6nt, i.e. hexamers, are identified assuming an equal probability for each word. The calculation of over-represented words shows that the hexamer TTTTTT is the most frequent word, followed by many more pyrimidine-rich words (as might have been expected from the composition of the polypyrimidine tract which is situated at

the end of introns). A graph showing the distribution of the hexamer TTTTTT shows a marked concentration at the end of introns, the position of the polypyrimine tract (Figure 31). In this example, however, the polypyrimidine tract is not considered. Instead another hexamer will be analysed: CTGACC. This word shows a high concentration just before the location of the polypyrimidine tract. It closely resembles the branch site (YTNAY, with Y being either C or T, N being any nucleotide). In the following it will be demonstrated, using the branch site as an example, how the clustering of over-represented words is applied to construct clusters of similar sequences and how weight matrices are produced from these clusters.

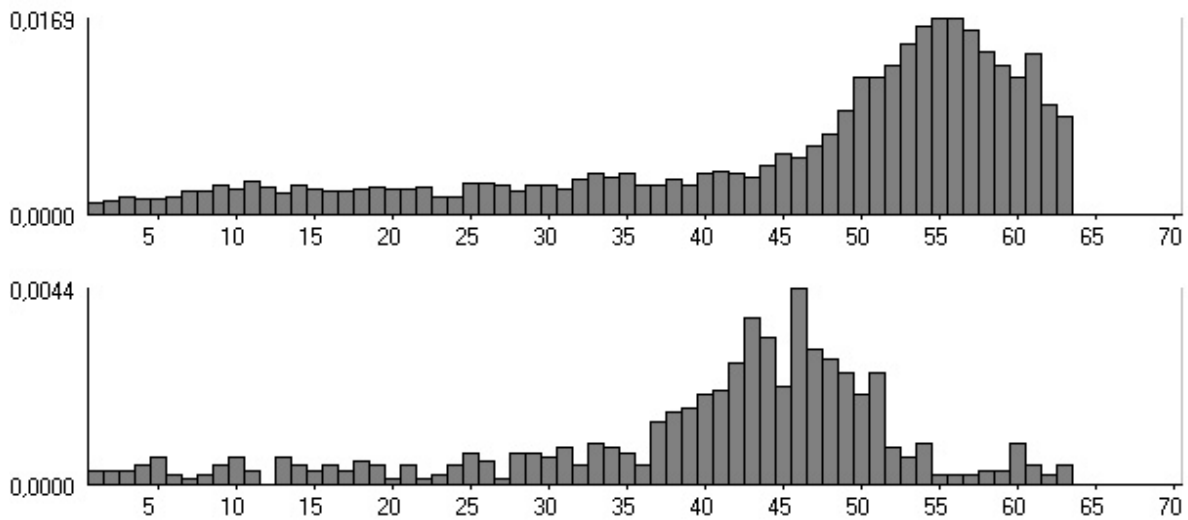



Figure 31. Occurrence of the hexamer TTTTTT at the end of human introns (top) and of the hexamer CTGACC (bottom), which resembles the branch point.

Analysis of the region in which most over-represented words are found

The previous analysis showed that the region where most CTGACC hexamers were found contains also many words which correspond to the polypyrimidine tract. In order to exclude high scoring words which represent the polypyrimidine tract, words will be calculated only for a very narrow range between -35 nt and -15 nt (range 35 to 55 in Figure 31) upstream of the acceptor splice site. The word CTGACC was most frequently observed in this region. After calculating over- and under-represented hexamers for this region (this analysis is shown in detail here, for details about the calculation of over-represented words refer to the previous chapter), words are clustered by pressing the button  or selecting *Statistics|Cluster over-/under-represented words* in the menu. The following dialog (Figure 32) offers several options: The *Number of words to cluster* allows to limit the clustering to the top ranking words only (according to the word list created for the calculation of over- and under represented words). In this example clustering will be conducted for the top 500 words, i.e. the 500 highest-scoring

words. The branch limit determines at which “distance” (here: sequence similarity), words are clustered into one cluster. A low branch limit will cluster only very similar words, while a large branch limit will create clusters with less similar words. For a first analysis a low branch limit should be used, which may be increased until the clusters show a reasonable composition. In this example a branch limit of 0.075 will be used. As mentioned before, the branch limit refers to the “distance” or similarity between words. Usually sequence identity will be used. This means that the “distance” refers to the fraction of nucleotides in a word which are not identical (other distances are also provided, but sequence identity will give appropriate results in most cases).

Detailed information about the neighbour joining tree on which the clustering is based can be displayed optionally (option: *Display tree information*), and a matrix of the distances between all pairs of words can be shown (option: *Display distance matrix*). Usually many clusters are formed during an analysis, therefore it is advisable to filter only the “best” clusters, e.g. those including the highest number of words (option: *Display clusters containing min. word number of*), those with a minimum cumulative score (i.e. the sum of the scores of all words; option: *Display clusters with a min. cumulative score of*), or clusters with a given minimum number of cumulative sequence hits (i.e. the sum of the sequence hits of each word; option: *Display clusters with min. cum. sequence hits of*). In this example, only clusters with a minimum cumulative number of 3000 sequence hits shall be displayed. Finally, the words are clustered by pressing *OK*.

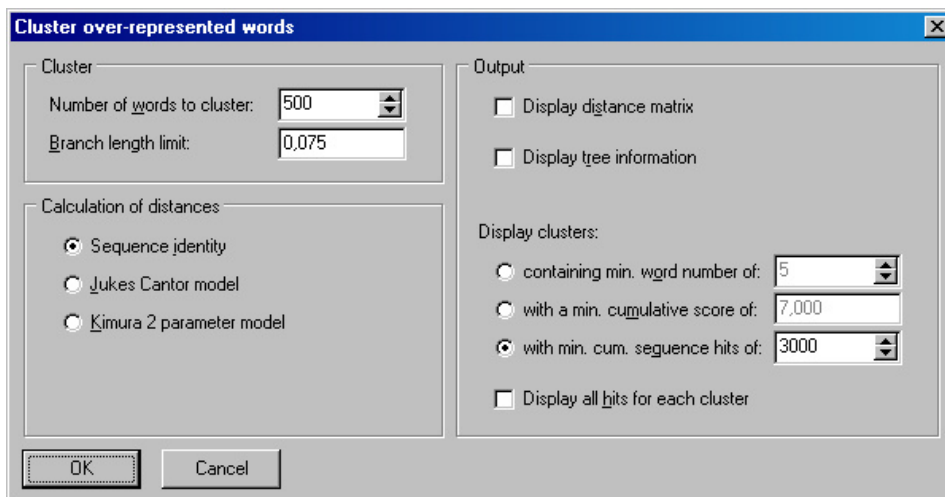


Figure 32. The dialog for clustering over- or under-represented words (short subsequences) for the calculation of consensus trees for putative signals.

Clusters and weight matrices

The output of this analysis indicates a total number of 77 clusters (Figure 33). Of these, only 6 clusters show at least 3000 cumulative sequence hits (i.e. the sum of the sequence hits of all words exceed 3000; however, the real number of hits of a cluster may be smaller than this number, because two words of one and the same cluster may be found in one and the same sequence).

```
Branch lengths of UPGMA tree of over-represented words:
```

```
Min. branch length: 0,0000  
Max. branch length: 0,1667  
Mean branch length: 0,0339
```

```
77 clusters found. Displaying clusters containing words found in at least 3000 sequences:
```

Figure 33. General information displayed for the clustering of over-/under-represented words.

Two of these 6 clusters two are displayed in Figure 34. These clusters resemble the human branch site YTNA. The words of both clusters are found in more than 3000 of all 10000 sequences, with a cumulative score of 13.1 and 16.1 respectively (Figure 34). For each cluster a simple weight matrix and a consensus sequence is displayed. The most conserved positions are marked with asterisks (****). Since the given weight matrices are based only on the words included in the clusters, the terminal positions of the matrices are represented only by very few words. For example, in cluster 5 the position 8 is only covered by 4 words, each showing another nucleotide. Consequently, the nucleotide frequencies at this position is 0.25 for each nucleotide. Evidently, 4 words represent a very small sample size. More accurate weight matrices can be calculated using the option *Display all hits for each cluster* (see Figure 32), which was not mentioned previously. When this option is activated hits are searched for each word in the cluster within the whole dataset. The resulting list of hits can be used as an alignment file for building a profile or weight matrix model (see chapter 5.1, copy all hits in a separate file and use this file as an alignment file for building the model).

Profiles for all six clusters observed in this example are shown in Figure 35. Four of these clusters seem to relate to the polypyrimidine tract since they show a high fraction of the nucleotides C and T (clusters 1, 3, 4 and 6). But the other two clusters, especially cluster 5, show a high similarity to the branch site YTNA. A profile for both combined clusters is displayed in Figure 36, as well as a profile of the human branch site from Senapathy et al. (1990) for comparison.

Cluster 2 (30 words):

```
--ttataa-
--aaataa-
---aataat
---aataaa
--taataa-
--taatgt-
-ttaatg--
-ctaatag--
tctaata---
--taatga-
-ataata--
-ttaata--
--taatat-
-taaatt--
-taaata--
ataaat---
--taattg-
--taattt-
-ataatt--
-ttaatt--
...
...
```

Cumulative score: 13,118; cumulative sequence hits:3205

Profile of cluster 2:

Pos	1	2	3	4	5	6	7	8	9
a	0,57	0,17	0,22	0,77	0,97	0,00	0,48	0,67	0,67
c	0,14	0,11	0,00	0,00	0,00	0,00	0,00	0,00	0,00
g	0,00	0,00	0,00	0,00	0,00	0,00	0,17	0,08	0,00
t	0,29	0,72	0,78	0,23	0,03	1,00	0,35	0,25	0,33

Cons: h h w w w t d d w

Cluster 5 (29 words):

```
-agtgac--
--gtgact-
-tgtgac--
---tgacca
--gtgacc-
---tgactg
-actgag--
---tgacct
-actgac--
--ctgacc-
-cctgac--
---tgaccc
-tctgac--
-gctgac--
--ctgact-
tgctga---
--ctgaca-
tcctga---
--ctgatt-
--ctgatg-
ggctga---
...
...
```

Cumulative score: 16,151; cumulative sequence hits:4373

Profile of cluster 5:

Pos	1	2	3	4	5	6	7	8	9
a	0,17	0,20	0,00	0,00	0,00	1,00	0,00	0,07	0,25
c	0,17	0,33	0,84	0,00	0,00	0,00	0,65	0,43	0,25
g	0,33	0,27	0,16	0,00	1,00	0,00	0,26	0,14	0,25
t	0,33	0,20	0,00	1,00	0,00	0,00	0,09	0,36	0,25

Cons: n n s t g a b n n

Figure 34. Two clusters which may represent the branch site YTNA upstream of human acceptor splice sites. Only the first words of the clusters are displayed here for demonstration.

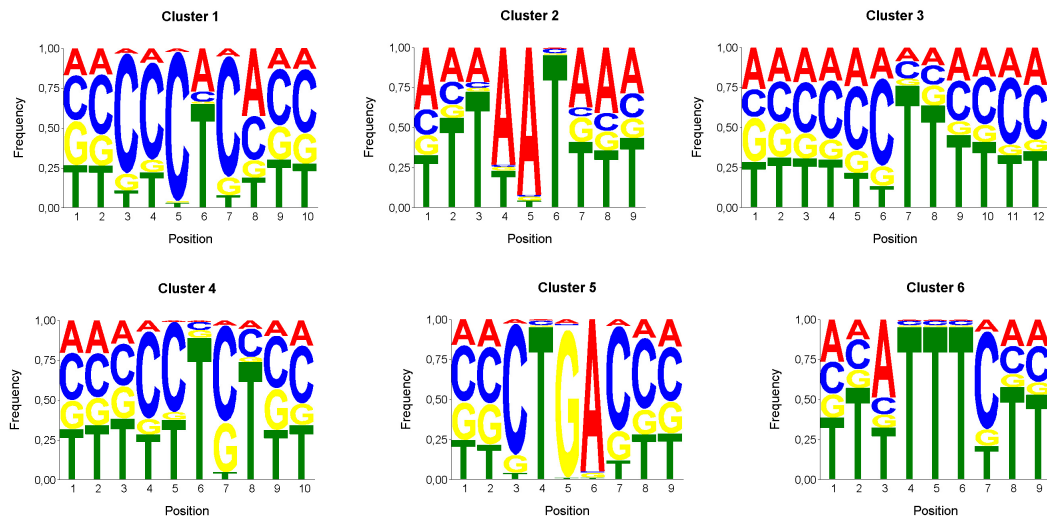


Figure 35. Profiles for six clusters of over-represented words.

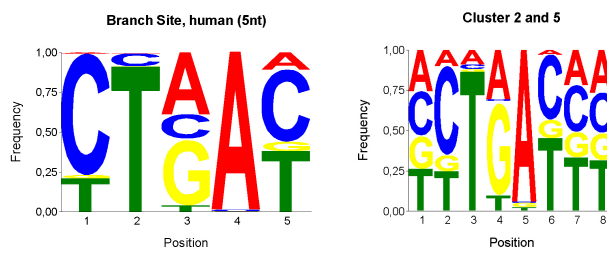



Figure 36. Profiles of the human branch site (left, according to Senapathy et al. 1990) and a calculated from the combined clusters 2 and 5.

4.5. Searching for patterns

4.5.1. Text search

Text search provides a simple way to search for patterns. In **Seqool** and **SeqoolM** text search is facilitated by the use of IUPAC nucleic acid codes, such as 'y' for pyrimidines, or 's' for strong bases (G or C). For example searching for the string 'yyyy' provides an easy way to identify most polypyrimidines.

4.5.1.1 Basic text search

Open the text search dialog by pressing the button  or by selecting *Search|Text search* in the menu. The text search dialog which opens is displayed in Figure 37. The easiest way to search for a string is to enter the search string in the left column of the table (column *Search sequence*; the column *Description* is only informative when searching with types). If IUPAC codes for nucleic acids are used, make sure to activate the option *Use IUPAC nucleic acid codes*. A list of IUPAC nucleic acid codes can be displayed by pressing the blue arrows next to the check box. Finally press *Search*. Subsequently hits are displayed in the *Graphics* page of the program (Figure 38).

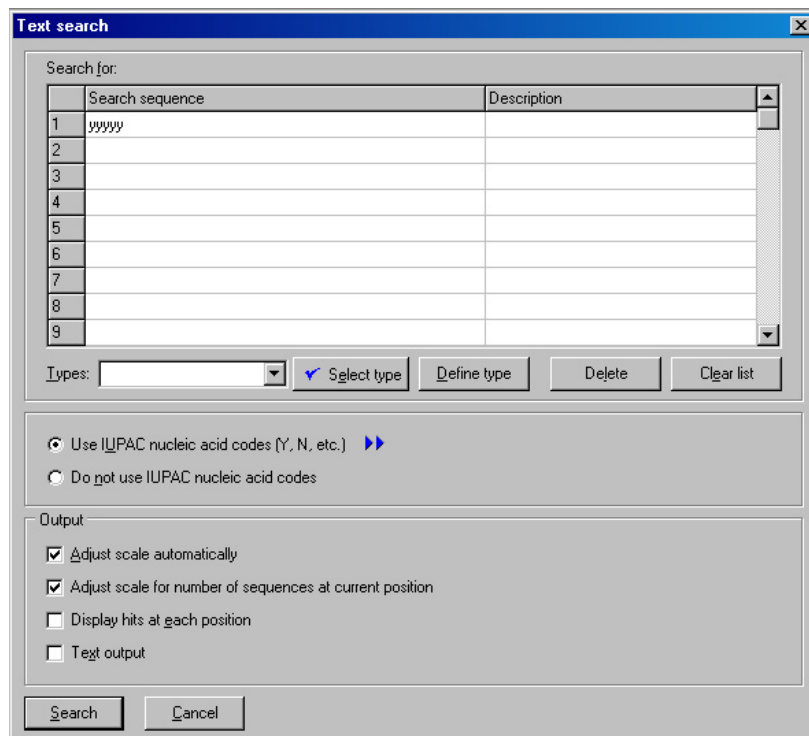


Figure 37. The text search dialog in **SeqoolM**.

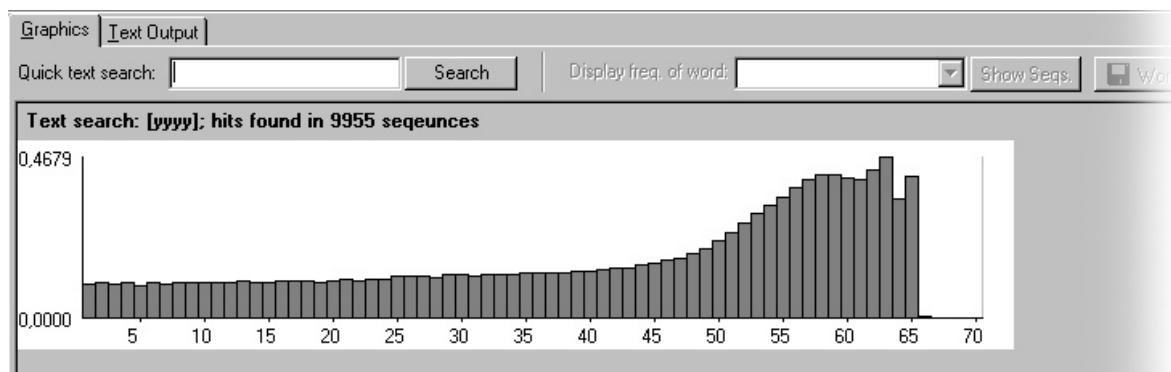


Figure 38. Hits are displayed in the hits graph.

Options

A few options allow to customize the output:

- | | |
|--|---|
| Adjust scale automatically | - Adjusts the scale of the graphic output automatically. If this option is deactivated, the y-axis shows the range from 0.0 to 1.0, otherwise the output is adjusted according to the highest frequency observed. |
| Adjust scale for number of sequences at current position | - Displays the frequencies of a word relative to the number of sequences in the dataset which cover the position where a word was found. This option is useful if a dataset contains sequences with varying lengths. For example, consider a dataset containing three sequences of varying lengths, e.g. 40 nt, 60 nt and 100 nt. If a given word is found at the 50th position in one of the three sequences the absolute frequency would be 0.333. However, only two sequences actually covered the position 50, hence the relative frequency is in fact 0.500. |
| Display hits at each position | - Shows the number of hits at each position of the sequences in the graphical output. |
| Text output | - Shows a table containing the number of observed and expected hits for each word, the number of sequences in which a word was found, the score, and the number of hits at each position in the sequences. |

Alternatively to the text search described above, a more rapid text search can be performed by entering a given search text in the box *Quick text search*, which is located at the top of the Graphics tab sheet (see Figure 38). The quick text search uses the same options currently selected in the Text search dialog (adjustment of scales, etc.).

4.5.1.2 Types – Frequently used / alternative search-patterns

SeqoolM offers the possibility to create *types* for searching. A *type* is intended for frequently used search patterns and it may include more than one search text. For example, the type 'Stop codon' allows to search for all three stop codons by including all three stop codons as alternative search patterns. For a detailed description of *types* and their use for text search refer to the respective section in the description of the program **Seqool** (chapter 3.4.1.3).

4.5.2. Searching with pattern recognition models

4.5.2.1 Introduction

The **Seqool** program package provides separate modules for creating a variety of pattern search models and models, e.g. position specific score matrices (weight matrices), profile hidden Markov models, and others, and more advanced models for combining basic models, such as decision trees or neural networks. Modules for building such models can be started by selecting the respective component in the *Tools* menu or using the respective tool buttons (Figure 39). All these models can be used for searching in **Seqool** and **SeqoolM**.

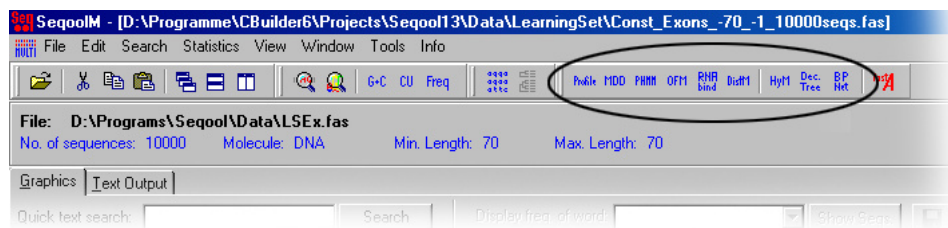




Figure 39. Tool buttons for starting program components for signal-recognition models.

4.5.2.2 Basic model search

Pressing the button  (or selecting *Search|Model search* in the menu) opens the model search dialog (Figure 40). When searching with a specific model, only models contained in the default model directory are listed. This default directory can be changed permanently in the main menu (menu *File|Configure*) or only temporarily (until exiting the program) by pressing the “Model directory” button (if a new model was saved in the model directory while the model search dialog was still open, the list of models can also be actualised by pressing ).

For selecting a model, choose one of the models listed in the *Select model* box and press the button *Add model*. Additional information about the model is displayed below the *Select model* box, such as the model type, the models name, the description, and the file name (see Figure 40). Press *Search* for starting the model search. An example of the graphical and the text output, respectively, is shown in Figure 41 and in Figure 42.

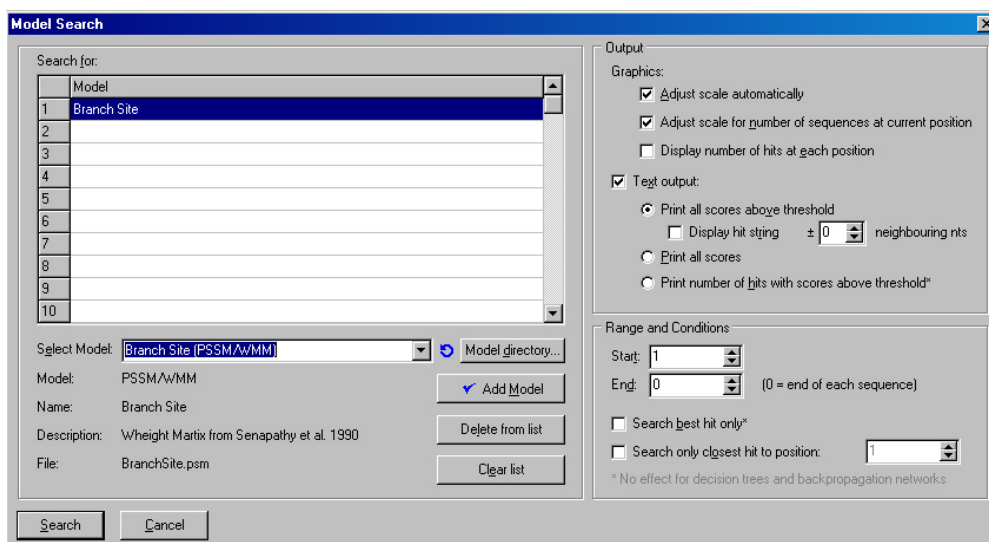


Figure 40. The model search dialog in *SeqoolM*.

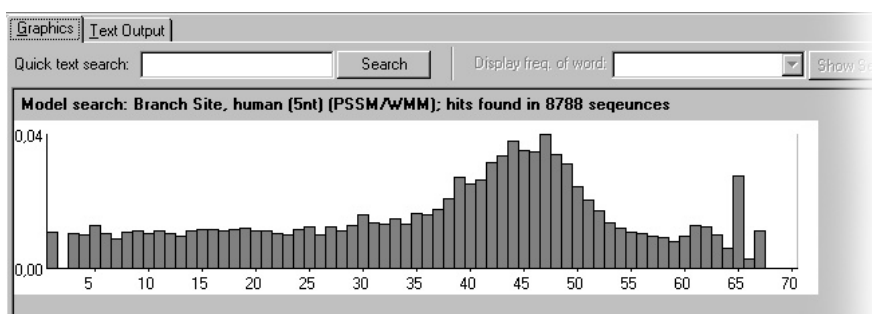


Figure 41. Example of search results for the model search.

```

Searching with PSMM/WMM: Branch Site, human (Wheight Martix from Senapathy et al. 1990)

Threshold: 3,0000

Seq.  Pos.  Score  Hit
1     48    3,300  aggccccaatatgat
2     39    6,981  aaagactgatcccca
3     38    7,048  tgtgctgacagctg
...
...
9999  47     5,667  tgtgtctcacttaga
10000 47     3,300  gaggcccaatatgca
10000 61     3,921  acctgttcactctgca

Mean hits per sequence: 1,92
Overall hits:           19170
Sequences:              10000

Hits at each position:
Pos.: 1    2    3    4    5    6    7    8    9    10   11   12   13   ...
Hits: 224  224  230  217  264  194  199  211  233  210  233  216  217  ...
    
```

Figure 42. Example of a text output for model search (options: *Print all scores above threshold* and *Display hit string* were activated).

Options

Several options allow to customize the graphical and the text output. A number of these options allow to refine or filter certain hits according to their score or their position. Furthermore, the sequence of all matches can be listed in the text output and may subsequently be used for a refinement of a model.

- Adjust scale automatically* - Adjusts the scale of the graphic output automatically. If this option is deactivated, the y-axis shows the range from 0.0 to 1.0, otherwise the output is adjusted according to the highest frequency observed.
- Adjust scale for number of sequences at current position* - Displays the frequencies of a word relative to the number of sequences in the dataset which cover the position where a word was found. This option is useful if a dataset contains sequences with varying lengths. For example, consider a dataset containing three sequences of varying lengths, e.g. 40 nt, 60 nt and 100 nt. If a given word is found at the 50th position in one of the three sequences the absolute frequency would be 0.333. However, only two sequences actually covered the position 50, hence the relative frequency is in fact 0.500.
- Display hits at each position* - Shows the number of hits at each position of the sequences in the graphical output.
- Text output* - Shows a tables with various information, depending on the options and filters selected. E.g. Scores and positions of the highest scoring hit for each sequence, all scores for each position of a sequence, etc. Several options are available for text output:
- Print all scores above threshold:* Displays the scores of each hit with a score that is higher than the score threshold of the model. For each hit the hit sequence can be displayed, including neighbouring sequences (option: *Display hit string*)
- Print all scores:* Displays the scores of each and every position within each sequence, irrespective of the score threshold of the model.
- Print number of hits with scores above threshold:* Shows the number of hits for each sequence which are above the score threshold of the model.

- Range* - Allows to search only within a given range, e.g. between the position 100 and 150 nt. Selecting 0 in the “End” box searches until to the end of each sequence.
- Search for best hit only* - Only the hit with the highest score is reported for each sequence.
- Search only closest hit to position* - Only the hit (with a score above the score threshold of the model) which is closest to a given position is reported.

4.6. Customizing SeqoolM

4.6.1. Configuring SeqoolM

Basic settings can be customized using the menu option *File|Configure...*. All options listed in the configure window (Figure 43) are permanent, i.e. the respective settings are loaded by default when the program starts. The *default type file* refers to the file containing all types which can be used for text search (see chapter 4.5.1.2). The *Codon usage file* defines which codon usage table is used for the calculation of codon usage of codon preference. The *Model directory* determines which folder contains the pattern recognition model which are available for searching in **SeqoolM** (when using a model comprising several submodels remember that also all sub-models must be located in the same folder as the main model). Finally, it can be selected how many recently opened files are displayed in the file menu.

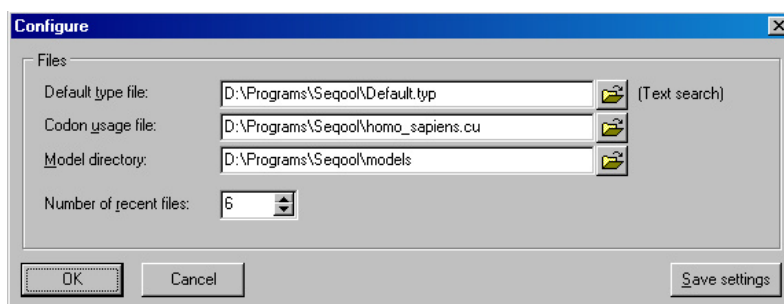


Figure 43. The dialog for customizing basic settings of **SeqoolM**.

4.6.2. Program priority

Since calculations may be time intensive for large sequence datasets, **SeqoolM** may occupy most computer resources (processor time). This may impede other applications to run smoothly. Therefore, the program priority of **SeqoolM** can be adjusted using the menu option *File|Configure...|Program priority*. Five settings can be selected. The highest priority setting (*highest*) may considerably slow down even the operating system, causing even mouse movements to be delayed. The two lowest priority levels (*low* and *idle*) allow other programs to run smoothly while calculations in **SeqoolM** will take more time than usually.

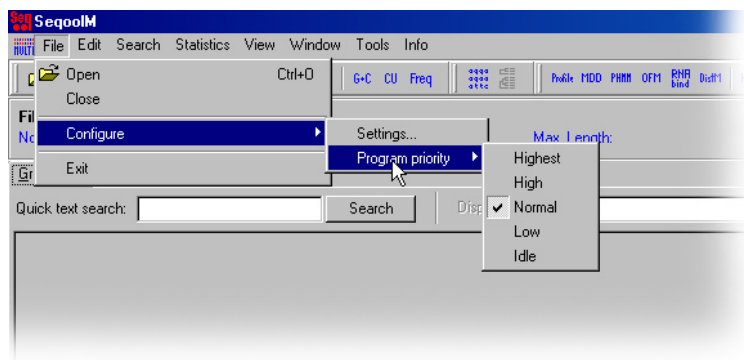


Figure 44. Program priority settings in *SeqoolM*.

5. Building pattern recognition models

5.1. Profiles (WMM, PSSM)

Module *Profile* 

5.1.1. Introduction

A simple way to identify signals in nucleotide sequences is to observe nucleotide frequencies at each position of signals and calculate probabilities for their occurrence. *Profiles* (or *weight matrices models*, WMM) show the probability of observing a nucleotide at a given position (Table 1). A graphical representation of this weight matrix is shown in Figure 46.

Position:	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	+1	+2
A	0,092	0,093	0,111	0,109	0,086	0,089	0,219	0,057	0,908	0,055	0,249	0,230
C	0,352	0,347	0,372	0,404	0,430	0,374	0,320	0,672	0,019	0,019	0,153	0,216
G	0,129	0,138	0,134	0,104	0,078	0,081	0,232	0,023	0,055	0,907	0,483	0,223
T	0,427	0,423	0,384	0,383	0,406	0,456	0,230	0,248	0,019	0,018	0,115	0,331

Table 1. Probabilities of observing a nucleotide at a given position around the human acceptor splice site, obtained from 5000 constitutive splice sites extracted from the Altextron database (Clark and Thanaraj 2002).

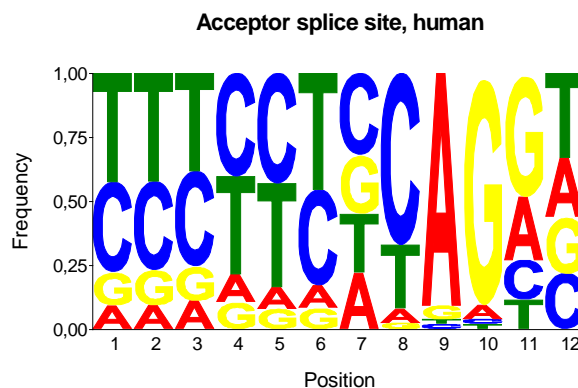


Figure 44. Graphical illustration of the weight matrix shown in Table 1.

Weight matrices can be used for searching for example for putative acceptor splice sites in unknown sequences by calculating the probability (more specifically the so-called log-odds score) of a subsequence to correspond to a splice site. Usually, a threshold is applied to distinguish putative signals from random or “false” signals. This threshold is determined

experimentally by comparing the distributions of scores obtained from the model for real splice sites and false splice sites (Figure 46).

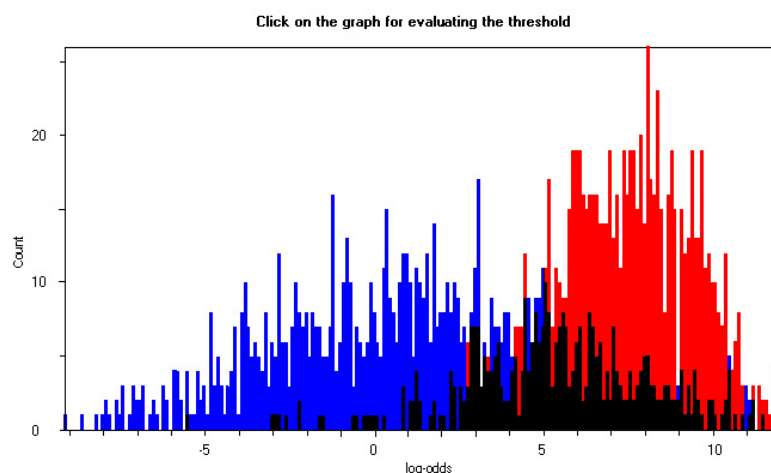


Figure 46. Score distribution of real human acceptor splice sites (red) and “false” acceptor splice sites (blue, subsequences containing ‘AG’ but not being real splice sites), obtained from a simple weight matrix model (profile).

Similar to weight matrix models are *position-specific score matrices* (PSSM). Position-specific score matrices differ from weight matrices only in the fact that nucleotide probabilities are transformed to log-odds scores. A PSSM of the preceding weight matrix model is given in Table 2.

Position:	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	+1	+2
A	-3,44	-3,43	-3,17	-3,20	-3,54	-3,49	-2,19	-4,14	-0,14	-4,18	-2,01	-2,12
C	-1,51	-1,53	-1,43	-1,31	-1,22	-1,42	-1,65	-0,57	-5,75	-5,69	-2,70	-2,21
G	-2,95	-2,86	-2,90	-3,26	-3,68	-3,62	-2,11	-5,43	-4,19	-0,14	-1,05	-2,16
T	-1,23	-1,24	-1,38	-1,38	-1,30	-1,13	-2,12	-2,01	-5,71	-5,78	-3,13	-1,60

Table 2. A position-specific score matrix (PSSM) of the human acceptor splice site.

5.1.2. Increased performance using information content

A modification introduced by Schneider (Schneider et al. 1986) applies information theory for searching signals. Information content is calculated for each nucleotide and each position, reflecting how much information of a signal is due to the occurrence of a specific nucleotide (see Schneider et al. 1986 and Schneider 1997). A model of the human acceptor splice site and using information content is shown in Figure 46. The size of nucleotide letters indicate the amount information content. Negative values are displayed by upside-down letters.

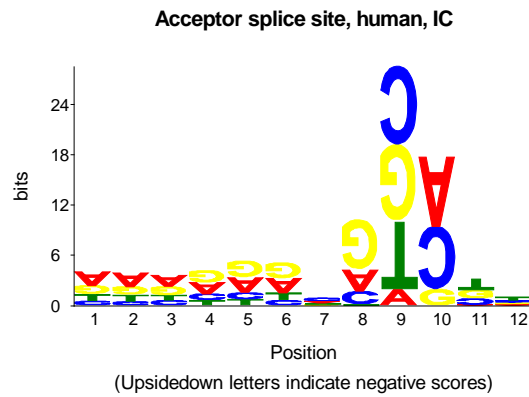


Figure 46. A model of the human acceptor splice site using information content (drawn using the module Profile).

5.1.3. Adding pseudocounts

If a profile is based on a small amount of sequences, it is probable that the obtained nucleotide frequencies are only very approximate. Consequently, searching with the corresponding model might not identify all real signals reliably. The recognition can be improved by adding *pseudocounts* to the obtained nucleotide frequencies. A number of different methods for the calculation of pseudocounts are proposed in the literature (see e.g. Durbin et al. 1998). In **Seqool**, pseudocounts are derived from PAM substitution matrices and can be adjusted using a pseudocount weight (as described in Durbin et al. 1998).

5.1.4. Higher order models

Profiles (or PSSMs) can also be calculated using di- or trinucleotide counts for each position. Notably, this procedure does not correspond to a Markov chain of order two or three, respectively. The inclusion of di- or trinucleotides might be useful when the nucleotide probabilities at neighbouring positions are mutually dependent.

5.1.5. Building a WMM/PSSM

Start the application: *Profile*

1. Select a Name for your model. Optionally add a short description.
2. Select an Alignment file containing a set of aligned sequences containing your signal.
3. Select a file containing Background frequencies (used for the calculation of pseudocounts). This file contains the expected nucleotide frequencies of the region where your signal is present. Frequency files can be created using **SeqoolM** (Menu-item: *Statistics|Oligonucleotides*).
4. Select if you want to calculate a conventional PSSM using Log-odd scores or Information content.
5. Select if you want to use nucleotide (order 1), dinucleotide (order 2) or trinucleotide counts (order 3) for you model. Usually, nucleotide counts do well.
6. If you want to calculate a PSSM using di- or trinucleotide counts and you do not have the respective background-frequency-file available (i.e. the "*.fre" file for di- or trinucleotides), you might select the option *Estimate di-/tri-nucleotide frequencies from nt frequencies*. In this case only a nucleotide frequency file is needed as background-frequency-file, and di- or trinucleotide frequencies are roughly estimated based on that file. However, it is recommended always to use a background frequency file.
7. If you have chosen to build a model calculating log-odds scores, select the Weight of pseudocounts. Models with small weights might not identify some real signals, while large weights will lead to unspecific models.
8. Select a Score threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
9. Select a Colour for the graphical representation of hits of this model in **Seqool**.
10. Press the Build button.

Evaluate the model using the following options:

Show parameters Display model parameters

Draw sequence logo Create a graphical representation of the model

Score sequences Score a set of sequences. Use this option to reveal the best score threshold by comparing score distributions of real signals and false signals (or random sequences) and for testing a model. Note that the sequences should have the same length as those of the alignment file used for training the model. For longer sequences additional nucleotides are ignored. The dialog offers some additional options, such as the saving of sequences above a score threshold.

5.2. Maximum Dependence Decomposition (MDD)

Module *MDD* 

5.2.1. Introduction

In several cases, nucleotide positions are mutually dependent, i.e. the probabilities of observing a nucleotide at one positions depends strongly on the nucleotides which are observed at other, even quite distant positions. Burge and Karlin (1997) developed *maximum dependence decomposition* (MDD) for the recognition of donor splice sites. MDD divides a set of signals into several subsets of more similar signals, and subsequently builds a weight matrix (WMM/PSSM) for each subset. This is done by repetition of the following steps:

First, the position is determined which (on average) correlates most with any other position in the signal (determined by a number of χ^2 -tests). Second, the set of analysed sequences is subdivided into those sequences containing the most frequent nucleotide of that position and those not containing it. Then, the preceding two steps are repeated for each subset.

The whole procedure is repeated until a minimum number of sequences is obtained in a subset. Finally, for each subset one weight matrix (WMM/PSSM) is built. When searching with an MDD model, a putative signal is first analysed regarding the nucleotides observed at the most significant positions, and then it is scored by the corresponding sub-model.

5.2.2. Example

The construction of an MDD model is demonstrated for a set of 5000 human donor splice sites (Table 3). First, a weight matrix is calculated from all sequences. Then, a consensus sequence is constructed from this weight matrix given a threshold of 30 percent. This means that all those bases are assigned to a letter in the consensus sequence which show a minimum frequency of 30 percent (0.3). The respective consensus in this example is [a/c]AGGT[a/g]AGTG (see Table 3). In the next step, all sequences are analysed at each position of the consensus sequence in order to reveal dependencies. This is done by many pairwise chi-square tests. The chi-square values indicate the level dependence between two positions. For example, position +3 shows the highest dependence with position +5, with the respective chi-square value being 412.7. The next highest dependence of position +3 is with position +4, with a chi-square of 230.9. To get an estimate which position shows the highest overall dependence, the single chi-square values are summed up for each position. In the example it becomes clear then, that position +5 shows the overall highest dependence (this position shows the highest sum of all

chi-squares: 1846.6). This means, that the overall composition of the donor site depends strongly on which base is present at that position.

After having identified the most significant position within the donor splice site, the set of splice sites are subdivided: Since the most frequent letter at position +5 is a G, the set of splice sites is split into two subsets, one containing those sequences which have a G at position +5, and another one with those sequences which have not. This is the first decomposition step. The resulting weight matrices for both subsets are shown in 2.a) and 2.b) of Table 3. Again, weight matrices are calculated for each subset separately, and the position with the highest dependence is identified by chi-square tests.

The complete MDD model construction in this example results in 8 different weight matrices. Figure 47 shows the score distributions for real acceptor splice sites and false splice sites (sequences containing AG, but not being real splice sites).

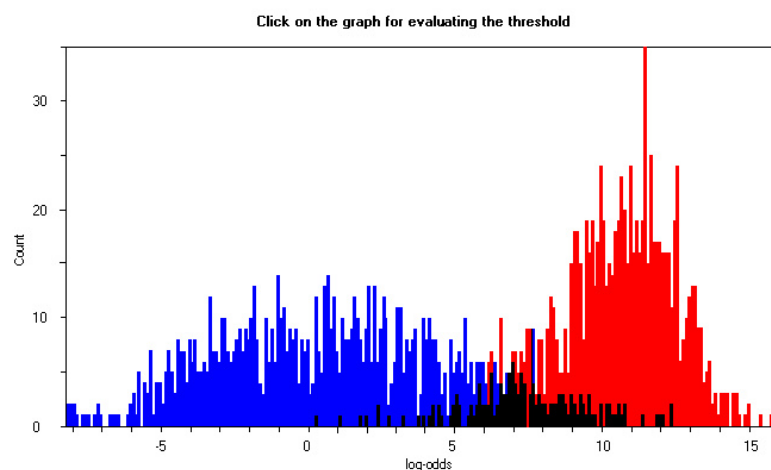


Figure 47. Score distributions of 1000 constitutive human acceptor splice sites (red) and 1000 false splice sites (blue, sequences containing AG, but not being real splice sites). The overall recognition is about 91 percent.

Note: When using a set of splice sites for building this model, make sure to use only canonical splice sites! If a set of donor splice sites contains e.g. few non-canonical splice sites, then the most significant positions determined by MDD will be the G and the T at the splice site core. The set will then be divided into two subsets, one of the sequences containing these bases (the canonical splice sites, i.e. almost all sequences) and a very small subset with sequences not containing these bases at the splice site core (the non-canonical splice sites). The latter set is likely to contain less sequences than the minimum number given by the subdivision threshold (e.g. a minimum of 200 sequences). Consequently MDD stops without subdividing anything!

1. Construction of a weight matrix from all sequences and analysis which positions show the strongest mutual dependence:

Weight matrix of all sequences

Pos	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10
A	0,27	0,28	0,25	0,27	0,29	0,26	0,28	0,28	0,26	0,28	0,28	0,29	0,33	0,63	0,10	0,00	0,00	0,56	0,70	0,08	0,17	0,28	0,21	0,20	0,21
C	0,27	0,26	0,28	0,27	0,26	0,26	0,25	0,24	0,26	0,25	0,25	0,28	0,37	0,12	0,03	0,00	0,00	0,03	0,08	0,06	0,16	0,20	0,26	0,28	0,24
G	0,26	0,23	0,26	0,25	0,23	0,26	0,26	0,23	0,25	0,24	0,21	0,22	0,18	0,12	0,79	1,00	0,00	0,38	0,11	0,80	0,20	0,32	0,25	0,25	0,27
T	0,21	0,23	0,21	0,21	0,23	0,22	0,22	0,25	0,23	0,24	0,25	0,21	0,12	0,13	0,07	0,00	1,00	0,02	0,11	0,06	0,47	0,21	0,28	0,27	0,29
Cons	-	-	-	-	-	-	-	-	-	-	-	-	a/c	A	G	G	T	a/g	A	G	T	G	-	-	-

Chi-square values for all pairs of positions:

Pos		-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10	Sum	
-15		-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-14		0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-13		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-12		0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-11		0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-10		0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-9		0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-8		0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-7		0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-6		0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-5		0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-4		0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-3	ac	0.3	2.5	5	2.5	4.5	3.7	1.5	11	2.5	6.1	28.1	162.7	-	141.3	27.9	0	0	14.2	45.7	64	39.3	8.5	3.6	4.3	3	582.3	
-2	A	6.4	14	1.5	1	36.1	2.5	7.9	20.6	10.7	3.1	28.1	13.5	417.1	-	197.3	0	0	92.5	209.5	294.3	181.3	41	2.9	25	7.9	1614.4	
-1	G	2.1	12.4	26.1	2.5	2.6	24.1	5.6	0.8	11.4	7.6	1.5	29.2	42.7	384.5	-	0	0	51.3	175.8	228.9	548.1	43.8	31.7	28.1	18.2	1679	
1	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0.1	
2	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0.1	
3	ag	6.9	2.2	1.4	0.3	2.8	7.8	7.9	1.4	5.7	1.4	8.2	1.8	20.4	63.7	50.6	0	0	-	35.6	20.2	18.2	6.7	5.3	3.6	4.4	276.5	
4	A	3.9	2.7	4.7	3.4	5	13.6	8	0.4	13.7	9.6	10.2	1.5	54.9	209.8	170.9	0	0	230.9	-	157.9	8.8	19.5	2.7	2.1	2.3	936.5	
5	G	9.1	4.4	3.5	0.9	10	4	14	7.1	15.6	4.9	29.3	11.4	65.8	296.7	227.9	0	0	412.7	433.4	-	130.3	99.3	18	24.8	23.4	1846.6	
6	T	4	3.9	18.6	6.6	2	13.4	0.3	5.2	15.6	5.5	1	8.1	47.1	184.4	539.6	0	0	46.7	20.2	122.6	-	131.5	43.7	11.2	29.8	1261.2	
7	G	23	9.3	32.6	5.3	9.8	10.3	9.3	16.1	10.7	6.1	28.8	20.9	14.1	17.7	0.8	0	0	83.9	31.3	42.6	160.5	-	71.5	54.3	26.7	685.6	
8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	
9		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	
10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	

Table 3. Maximum dependence decomposition (human donor sites, -15..+10 nt):

2. Construction of weight matrices for all those sequences containing or not containing G at position +5:

a) Decomposition of those sequences containing G at position +5

Weight matrix of the sequences containing G at position +5

Pos	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10
A	0.26	0.28	0.24	0.26	0.28	0.25	0.27	0.28	0.25	0.27	0.27	0.29	0.32	0.57	0.12	0.00	0.00	0.49	0.74	0.00	0.16	0.29	0.20	0.19	0.20
C	0.28	0.27	0.28	0.27	0.26	0.26	0.26	0.25	0.27	0.24	0.26	0.28	0.36	0.14	0.04	0.00	0.00	0.04	0.04	0.00	0.16	0.20	0.26	0.29	0.25
G	0.26	0.23	0.27	0.26	0.23	0.26	0.26	0.24	0.26	0.24	0.22	0.22	0.19	0.14	0.75	1.00	0.00	0.45	0.12	1.00	0.18	0.33	0.26	0.26	0.28
T	0.20	0.22	0.21	0.21	0.22	0.22	0.21	0.24	0.23	0.24	0.25	0.21	0.13	0.15	0.09	0.00	1.00	0.03	0.10	0.00	0.50	0.18	0.27	0.26	0.27
Cons	-	-	-	-	-	-	-	-	-	-	-	-	ac	A	G	G	T	ag	A	G	T	G	-	-	-

Chi-square values for all pairs of positions:

Pos		-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10	Sum	
-15		-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-14		0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-13		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-12		0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-11		0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-10		0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-9		0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-8		0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-7		0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-6		0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-5		0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-4		0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-3	ac	0.2	3.4	2.8	1.5	4.5	5.1	2.5	7.9	1.8	4.6	26.1	131.8	-	101	13.8	0	0	13.3	22.2	0	30	8	1.6	3.4	2.3	388	
-2	A	4.3	13.3	3.6	1.9	32	4.3	5.2	15.6	10.3	3	13.1	11.2	319.5	-	94	0	0	87.6	171.7	0	138.3	25.3	3.1	18.5	5.3	981.2	
-1	G	1.2	10.6	31.9	2.6	3.6	29.3	2.3	0.1	20.9	6.7	1.4	36.3	25.7	260.3	-	0	0	87.1	144.6	0	491.8	38.1	40.2	30.8	24	1289.4	
1	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0.1	
2	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0.1	
3	ag	5.3	3	0.1	0.6	4.2	6.4	10.8	2.5	7.7	0.7	8.1	1.6	22.2	85	63.3	0	0	-	32	0	13.9	4.7	8.7	4.3	3.4	288.2	
4	A	5.5	2.9	3.3	1.8	3.9	12.7	6.5	0.4	8.5	5.5	8.3	0.6	34	159.5	140.8	0	0	186.2	-	0	0.4	5.3	4.9	1	3.2	595.5	
5	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0.1	
6	T	1.5	5	25.8	7	1.5	14.7	0.3	2.3	17.2	2.5	4.2	11.6	36.6	146.1	487	0	0	23.3	1.6	0	-	107.2	42.7	14	33.1	985.1	
7	G	11.3	7.9	23.9	3.1	12.2	3.7	8.4	12	4.6	2.7	28.7	14.2	12.5	8.2	0.4	0	0	54.5	17.6	0	117	-	42.3	39.1	15.2	439.2	
8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0.1	
9		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0.1
10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0.1

Table 3. Continued.

b). Decomposition of those sequence not containing G at position +5:

Weight matrix of the sequences not containing G at position +5

Pos	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10
A	0.29	0.30	0.27	0.27	0.31	0.28	0.29	0.27	0.30	0.30	0.32	0.32	0.37	0.86	0.02	0.00	0.00	0.84	0.55	0.39	0.23	0.24	0.25	0.24	0.23
C	0.25	0.24	0.28	0.27	0.25	0.25	0.24	0.23	0.24	0.26	0.22	0.25	0.42	0.04	0.00	0.00	0.00	0.01	0.22	0.29	0.18	0.19	0.25	0.24	0.21
G	0.24	0.22	0.25	0.24	0.19	0.25	0.22	0.22	0.22	0.22	0.17	0.20	0.15	0.04	0.96	1.00	0.00	0.13	0.08	0.00	0.28	0.25	0.21	0.22	0.23
T	0.22	0.24	0.20	0.22	0.25	0.22	0.25	0.28	0.24	0.22	0.29	0.23	0.06	0.06	0.01	0.00	1.00	0.02	0.15	0.32	0.31	0.32	0.29	0.31	0.33
Cons	-	-	-	-	A	-	-	-	-	A	A	A	a/c	A	G	G	T	A	A	a/t	T	T	-	T	T

Chi-square values for all pairs of positions:

Pos	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6	7	8	9	10	Sum	
-15	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-14	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-13	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-12	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-11	A	1.5	2.8	7.4	55.9	-	14.4	1.8	16.3	0.4	2.8	8.7	8.4	5.8	4.8	8.8	0	0	6.2	1.1	8.7	0.4	1.8	2.1	5.8	1	166.7
-10	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-9	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-8	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-7	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
-6	A	2	7.6	3.9	6	2.7	0.8	5.5	4	18	-	17.1	1.7	0.1	4.9	0.2	0	0	1.5	0.8	2.8	1.4	0.8	1.8	3.7	0.1	87.6
-5	A	0.6	3.3	11.3	2.1	5.2	4.1	2	1.4	0.9	24.8	-	1.2	2.2	6.6	5.9	0	0	5	3.4	3.3	0.7	1.5	6.5	0.7	2.1	94.7
-4	A	0.8	2.1	7.6	4.1	3.3	1.7	4	1.2	8.7	10.2	12.7	-	17.4	7.1	8.8	0	0	1.5	6.9	2.3	1.6	8.1	5	2.1	9.4	126.5
-3	ac	0.8	0.7	3.9	8.3	0.5	2	0.9	8.4	0.5	3.1	5.5	45.6	-	7.9	4.3	0	0	7.1	11.5	7.6	3.6	3.2	3.2	0.5	3	131.9
-2	A	6.5	4.1	6.1	1.1	3.7	3.9	0.8	3.6	2.9	2.8	12.5	7.9	53.9	-	59.6	0	0	4.4	2.6	0.3	4.8	1	2.5	7.9	0.7	193.7
-1	G	0.5	7.8	0.2	2.3	8	0.7	0.4	0.7	1.1	1.2	18.7	8.2	7.9	44.7	-	0	0	2.9	23.7	6.9	19.4	21.9	9.7	4.3	3	194.1
1	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0.1
2	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0.1
3	A	13.4	2.7	14.9	2.6	1.1	5	1.7	0.7	2.6	3.3	8.1	4.7	6.8	7.7	3.5	0	0	-	21.3	5.7	0.1	20	3.5	5.8	7.2	142.4
4	A	1.5	1.9	1.5	1.5	1.4	3.1	0.3	1	14.1	3.8	4.6	2.3	7.2	9.1	23.4	0	0	12.4	-	12.4	4	14.8	1.5	9	10.5	141.4
5	at	3.6	0.4	8.3	0.6	5.2	6.9	4.5	12.7	7.9	3.8	16.1	2.5	4.5	0.5	32.9	0	0	10.4	8.5	-	89.4	20.3	20.9	9.5	29.1	298.6
6	T	9.3	0.4	1	5.1	6.8	1.7	4.2	5.3	4.4	4.5	0.8	3.8	1.1	1.6	4.5	0	0	0.2	6.3	3.1	-	17.6	18.9	1.8	7.9	110.4
7	T	7.8	4.8	6.4	3.9	4.3	18.4	0.2	3.5	9.9	2.1	0.3	7.4	9.2	1.3	30.8	0	0	2.9	22.1	2.4	4.6	-	21.2	16.6	4.3	184.4
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0.1
9	T	1.9	9.6	1.6	11.3	3.7	1.4	3.1	1	1.7	6.3	3.6	4.3	1.2	9.1	4.8	0	0	1.3	12.8	1.9	5.5	17.7	27.5	-	5.6	136.9
10	T	4.6	3.5	3.6	5.2	1.8	8.6	4.1	4.3	10.6	9.9	2.4	2.1	6.6	3	5.5	0	0	1.7	13.2	5.5	5	16	7.2	22.7	-	147.1

Table 3. Continued.

5.2.3. Building an MDD model

Note: All single submodel files (WMMs/PSSMs) used for an MDD model must be located in the same file path.

Start the application: *MDD*

1. Select a Name for your model. Optionally add a short description.
2. Select an Alignment file containing a set of aligned sequences containing your signal.
3. Select a file containing Background frequencies (used for the calculation of pseudocounts). This file contains the expected nucleotide frequencies of the region where your signal is present. Frequency files can be created using **SeqoolM** (Menu-item: *Statistics\Oligonucleotides*).
4. Select if you want to use conventional PSSMs using Log-odd scores or Information content.
5. Select a Cutoff value (nucleotide percentage) for the calculation of the consensus (30 percent should be good in most cases).
6. Select the Significance level for the χ^2 -tests (0.001 should be good for most cases).
7. Define when to stop the iterative decomposition of the sequence set, i.e. at which *minimum number of sequences* in a subset. Too low numbers will result in the subdivision into too many subsets with too few sequences for each subset, which results in an unreliable model.
8. If you have chosen to build a model calculating log-odds scores, select the Weight of pseudocounts. Models with small weights might not identify some real signals, while large weights will lead to unspecific models.
9. Select a Score threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
10. Select a Colour for the graphical representation of hits of this model in **Seqool**.
11. Press the Run MDD button.

Evaluate the model using the following options:

<i>Show decision tree</i>	Display matrices and other results
<i>Score sequences</i>	Score a set of sequences. Use this option to reveal the best score threshold by comparing score distributions of real signals and false signals (or random sequences) and for testing a model. Note that the sequences should have the same length as those of the alignment file used for training the model. For longer sequences additional nucleotides are ignored. The dialog offers some additional options, such as the saving of sequences above a score threshold.

5.3. Profile Hidden Markov Models (PHMM)

Module *PHMM* 

5.3.1. Introduction

Profile hidden Markov models are similar to Weight matrices, since they measure nucleotide frequencies for each position of a signal, but they additionally include the possibility that some residues are missing or that residues are inserted into a signal. Since most signals on DNA or RNA are uninterrupted short sequences, e.g. snRNAs of the splicing complex, simple matrices are adequate for their identification. However, in some cases binding mechanism involve more than one recognition motive. For example in the polypyrimidine-tract-binding proteins two RNA recognition motives were proposed (Banerjee et al. 2003). When binding to the polypyrimidine tract, two short subsequences are recognized, which are separated by a variable stretch of unbound RNA.

5.3.2. Example

For illustration of how profile HMMs are built, a simple alignment of fictive sequences is used:

	1	2	3	4	5	6	7	8	9
Sequence 1	a	c	a	-	-	a	t	g	
Sequence 2	t	c	a	a	c	t	a	t	a
Sequence 3	t	c	a	a	c	t	a	t	a
Sequence 4	a	c	a	c	-	-	a	g	c
Sequence 5	a	g	a	-	-	a	t	c	
Sequence 6	a	c	c	g	-	-	a	t	c

A weight matrix from this alignment contains the following probabilities:

Pos	1	2	3	4	5	6	7	8	9
A	0.66	0.00	0.83	0.50	0.00	0.00	1.00	0.00	0.00
C	0.00	0.83	0.00	0.25	1.00	0.00	0.00	0.00	0.83
G	0.00	0.17	0.17	0.25	0.00	0.00	0.00	0.17	0.17
T	0.33	0.00	0.00	0.00	0.00	1.00	0.00	0.83	0.00

Notably, this weight matrix does not include any information about the inserted bases (inserts) observed in the sequences 2 and 3 (inserted CT), and about the missing bases in the sequences 1 and 5 (missing base at position 4). In reality, it is not really evident which bases are missing or which ones are inserted. It could be argued that sequences 2, 3, 4, and 6 have inserts, but of variable length (no sequences would show a missing base then). In order to analyse this alignment more systematically, those positions which are covered in the majority of

all sequences are assumed to reflect the “original” signal (applying a minimum percentage of 50%). These positions are labelled “matches” (marked with an M):

```

123456789
aca---atg
tcaactatc
tcaactatc
acac--agc
aga---atc
accg--atc
MMMM MMM
    
```

Given these labels, two sequences of this alignment contain an insert beginning after position 4 (2 of the 4 sequences which have a base at position four = 50 %). In these two sequences, the insert is followed by another insert in one case, while the next time, the insert ends (an insert of two nucleotides). That means, that once the insert “starts”, the probability “continuing” the insert is 50 %. Apart from the two sequences containing the insert, 2 sequences of all 6 lack a base at position 4 (33 %). Referring again to the defined “matches”, the probability of continuing with a second missing base (a “delete”) is 0. Since these sequences show a base at the next “match” position (position 7), the probability of continuing with a “match-state” is 1.

Combining this information about “deletes” and “inserts” with the weight matrix, the following graphical representation of the profile HMM results (Plan 9 architecture, Figure 48). Deletes are represented by circles at the top. Inserts are depicted as diamonds in the middle of the graph. Matches are shown as rectangles at the bottom of the graph. They also contain the probabilities of the weight matrix. Lines connecting these match-, insert, and delete “states” correspond to the probabilities of observing a transition from one state (match, insert, delete) to another.

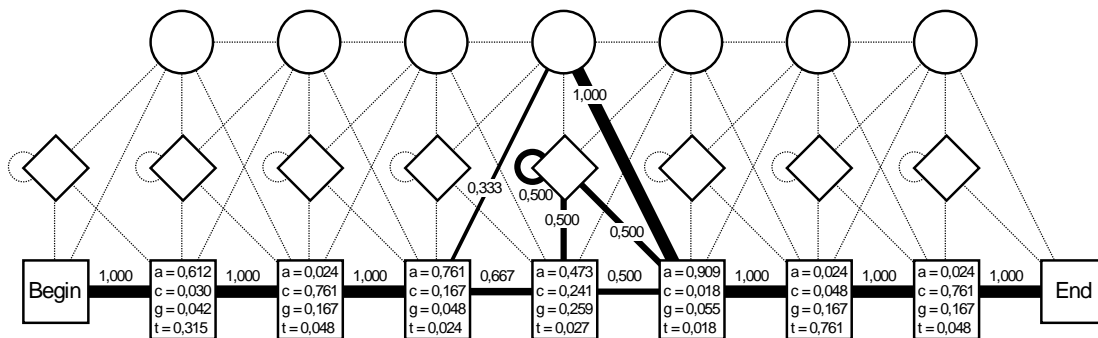


Figure 48. A simple example of a profile HMM (drawn using the module PHMM).

When searching for a signals within an unknown sequence, a putative hit sequence is “reconstructed” by the HMM by trying to find a way (“path”) through the HMM, which corresponds to the sequence. Due to inserts and deletes, an identical sequence can usually be generated by a number of different paths through the graph (except in the simple example given above). For scoring a sequence, either the score of the most probable path is calculated (Viterbi-algorithm), or the cumulative score of all possible paths (Forward-algorithm). The Plan 9 profile HMM of **Seqool** uses the latter method.

As in profiles models, pseudocounts calculated from substitution matrices can be added also to profile HMMs, and models can be built using either mono-, di-, or trinucleotides (order).

5.3.3. Building a PHMM

Start the application: *PHMM*

1. Select a Name for your model. Optionally add a short description.
2. Select an Alignment file containing a set of aligned sequences containing your signal.
3. Select a file containing Background frequencies (used for the calculation of pseudocounts). This file contains the expected nucleotide frequencies of the region where your signal is present. Frequency files can be created using **SeqoolM** (Menu-item: *Statistics\Oligonucleotides*).
4. Select if you want to use nucleotide (order 1), dinucleotide (order 2) or trinucleotide counts (order 3) for you model. Usually, nucleotide counts do well.
5. If you want to calculate a PHMM using di- or trinucleotide counts and you do not have the respective background-frequency-file available (i.e. the “*.fre” file for di- or trinucleotides), you might select the option *Estimate di-/tri-nucleotide frequencies from nt frequencies*. In this case only a nucleotide frequency file is needed as background-frequency-file, and di- or trinucleotide frequencies are roughly estimated based on that file. However, it is recommended always to use a background frequency file.
6. Define match-states: Either define matches automatically, choose a *Insert cutoff value*, e.g. define a match if no more than 50 percent of all sequences show a missing base at a position, or select match-states manually by using the option *Select match-states*.
7. Optionally select *Model insert emissions with from frequencies from alignment*. In this case, the probability of observing each base in a given insert is calculated from the observed nucleotide frequency of that insert.
8. Select the Weight of pseudocounts. Models with small weights might not identify some real signals, while large weights will lead to unspecific models.
9. Select a Score threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
10. Select a Colour for the graphical representation of hits of this model in **Seqool**.
11. Press the Build button.

Evaluate the model using the following options:

<i>Show parameters</i>	Display model parameters
<i>Draw HMM</i>	Create a graphical representation of the model
<i>Score sequences</i>	Score a set of sequences. Use this option to compare score distributions of real signals and random or false sequences, for evaluating score thresholds, and for testing a model. Note that the sequences should have the same length as those of the alignment file used for training the model. For longer sequences additional nucleotides are ignored. The dialog offers some additional options, such as the saving of sequences above a score threshold.
<i>Show Matrix for Seqs...</i>	Score a set of sequences and display the matrices displaying the paths through and scores of the HMM.

5.4. Oligonucleotide-frequency-models (OFM)

Module *OFM* 

5.4.1. Introduction

Oligonucleotide-frequency-models (OFMs) score the composition of a sequence, based on the frequencies of oligonucleotides (note that the module DefOFM additionally allows the use of GC content, codon usage, or codon preference instead). OFMs were introduced by Wang and Marin (2006) who applied them in combination with profiles and other models for the classification of constitutive and cryptic/alternative splice sites. When a signal is not characterized by a specific recognition motive alone, but also by a certain oligonucleotide distribution (e.g. due to the presence of regulatory elements, or due to codon usage), then addition of an OFM to a classical signal search model can increase the overall recognition of the signal. In constitutive acceptor splice sites, for example, splice site recognition can be enhanced by combining a profile or PSSM model for the core splice site with an OFM model for the downstream exon, and another OFM of the upstream intron. These OFMs can model the different oligonucleotide distributions in exons and introns. When using such a combination for searching splice sites within unknown sequences, splice site recognition will be based on both, the detection of a strong core splice site and the occurrence of downstream and upstream stretches with oligonucleotide compositions which are typical to introns or exon, respectively.

During the training process of OFMs, oligonucleotide frequencies are first calculated from a training set. After all frequencies are known, a given test-sequence can be scored using the following formula:

$$score = \sum \log_2 \left(\frac{p_{observed}}{p_{background}} \right)$$

where $p_{observed}$ is the observed frequency of a given oligonucleotide and $p_{background}$ is the expected or background frequency of that nucleotide.

Besides OFMs, this program module includes other sequence composition statistics, such as codon usage or GC content.

5.4.2. Example

The following example demonstrates the calculation using an OFM for the first 24 nt of (constitutive) human exons. Trinucleotides will be used in this example. Table 4 lists the frequencies of trinucleotides observed in the exon start region (1..24 nt) and the frequencies for whole exons (background frequencies).

	Frequency observed	Background frequency ^a
aaa	0.0206	0.0155
aac	0.0146	0.0131
aag	0.0212	0.0239
aat	0.0142	0.0091
aca	0.0171	0.0163
acc	0.0156	0.0186
acg	0.0086	0.0079
act	0.0142	0.0133
etc.	etc.	etc.

Table 4. Trinucleotide frequencies for the first 24 nt of constitutive human exons. ^a Trinucleotide frequencies from Buset and Guigo (1996).

Based on these frequencies the score of the sequence AAACAAT is calculated (this sequence contains the trinucleotides AAA, AAC, ACA, CAA, and AAT):

$$\begin{aligned}
 score &= \log_2 \left(\frac{p_o(aaa)}{p_b(aaa)} \right) + \log_2 \left(\frac{p_o(aac)}{p_b(aac)} \right) + \log_2 \left(\frac{p_o(aca)}{p_b(aca)} \right) + \log_2 \left(\frac{p_o(caa)}{p_b(caa)} \right) + \log_2 \left(\frac{p_o(aat)}{p_b(aat)} \right) \\
 &= \log_2 \left(\frac{0.0206}{0.0155} \right) + \log_2 \left(\frac{0.0146}{0.0131} \right) + \log_2 \left(\frac{0.0171}{0.0163} \right) + \log_2 \left(\frac{0.0177}{0.0198} \right) + \log_2 \left(\frac{0.0142}{0.0091} \right) \\
 &= 1.1083
 \end{aligned}$$

The positive score indicates that the sequence AAACAAT shows a trinucleotide composition which is more frequent in the exon start region (first 24 nt of an exon) than in average exon sequences.

5.4.3. Adding Pseudocounts

Especially for long oligonucleotides frequencies are difficult to estimate reliably using a limited number of training sequences. For example, the use of oligonucleotides of length six results in $4^6 = 4096$ different hexamers whose frequencies have to be estimated from a training set. It is

likely that some of the hexamers are rarely or even never observed. In such cases, it is strongly recommended to include pseudocounts in the model, otherwise sequences containing very rare oligonucleotides are incorrectly scored. As for the previous model types, pseudocounts of OFMs are derived from PAM substitution matrices and can be adjusted using a pseudocount weight (see Durbin et al. 1998).

5.4.4. Building an OFM

Start the application: *OFM*

1. Select a Name for your model. Optionally add a short description.
2. Select an Alignment file containing a set of aligned sequences containing your signal (only necessary for calculation of oligonucleotide frequencies).
3. Select a file containing Background frequencies (only necessary for calculation of oligonucleotide frequencies; used for the calculation of pseudocounts). This file contains the expected nucleotide frequencies of the region where your signal is present. Frequency files can be created using **SeqoolM** (Menu-item: *Statistics|Oligonucleotides*).
4. Select if you want to use score Oligonucleotide frequencies, GC content, Codon usage, or Codon preference.
5. If you selected oligonucleotide frequencies:
 - Choose the Length of the oligonucleotides to score
 - Select a Pseudocount weight. Models with small weights might not identify some real signals, while large weights will lead to unspecific models. Alternatively you may define a default frequency for oligonucleotides ("words") which were never found in your set of sequences. This makes the calculation much faster. For hexanucleotides, the calculation of pseudocounts with a given weight can take 15 minutes. If you don't want to wait, just kill the program and use default frequencies for missing oligonucleotides.
 - If you want to calculate an OFM using oligo-nucleotides and you do not have the respective background-frequency-file available (i.e. the "*.fre" file), you might select the option Estimate oligonucleotide frequencies from nt frequencies. In this case only a nucleotide frequency file is needed as background-frequency-file, and oligonucleotide frequencies are roughly estimated based on that file. However, it is recommended always to use a background frequency file.

If GC content, codon usage, or codon preference was selected:

- Select the Length of your model, i.e. for how many bases the model shall be calculated.
 - Optionally select to score only the best reading frame. This might be useful for scoring e.g. exon codon usage. If this option is selected, only the highest of all three scores (of the three reading frames) will be reported.
 - Optionally choose a file containing the codon usage and codon preference information for a specific organism. The default setting uses human data.
6. Select a Score threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
 7. Select a Colour for the graphical representation of hits of this model in **Seqool**.
 8. Press the Build button.

Evaluate the model using the following options:

Show parameters

Display model parameters

Score sequences

Score a set of sequences. Use this option to compare score distributions of real signals and random or false sequences, and for evaluating score thresholds. The corresponding dialog offers some options, such as the saving of sequences above a score threshold.

5.5. An RNA binding model based on binding energy (RNABind)

Module *RNABind* 

5.5.1. Introduction

This experimental model calculates the binding energy for a given nucleotide sequence and a RNA- or DNA-binding motif, not taking the formation of loops or bulges into account. This model is appropriate for short binding signals, e.g. snRNA binding signals, where the formation of loops is unlikely. The model uses default binding energies, listed below. However, energies can be selected by the user. The overall binding energy is calculated by the sum of the energies of all single bases.

	A	C	G	T
A	0	0	0	-2
C	0	0	-3	0
G	0	-3	0	-1
T	-2	0	-1	0

Table 5. Default binding energies used in RNABind.

5.5.2. Example

The donor splice site is recognized by the factor U1 snRNP in the first step of splicing. This recognition is caused by base-pairing of a short stretch of the snRNA-part with the splice site (positions -2..+6). In human the sequence of this signal is TCCATTCA (or UCCAUUCA, respectively). The calculation of the binding energy for a fictive test-sequence is illustrated:

U1 snRNA binding motive:	t	c	c	a	t	t	c	a
Test-sequence:	a	g	a	t	g	a	c	c
Binding energy for each base pair:	-2	-3	0	-2	-1	-2	0	0
Total binding energy:	-10							

When scoring real constitutive donor sites and false donor splice sites (i.e. sequences found near real donor splice sites which contain a GT dinucleotide, but are not real constitutive splice sites), two partially overlapping score distributions are observed (Figure 49). Both distributions are separated at an energy-threshold of about 12.5. Using this threshold, about 90 percent of true and false donor constitutive splice sites are classified correctly.

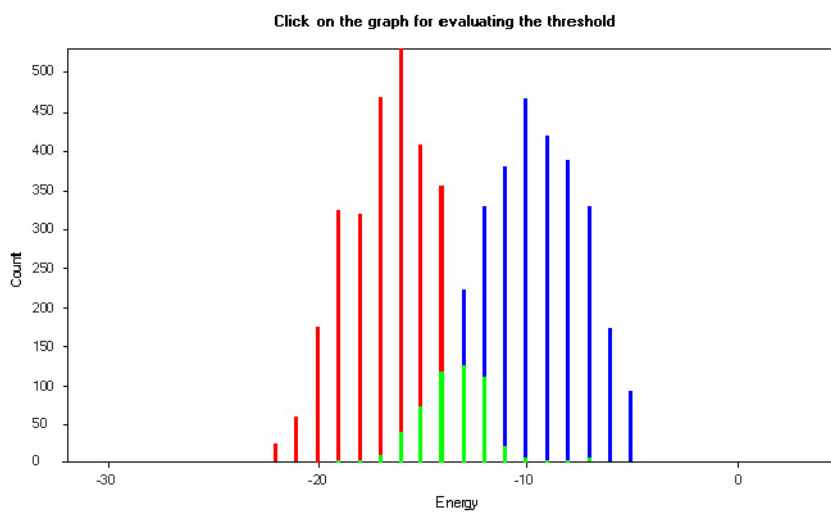


Figure 49. Score distributions of real (constitutive) donor splice sites and false splice sites (sequences found near real donor splice sites containing a GT dinucleotide, but not being confirmed constitutive splice sites). Blue - Confirmed constitutive donor sites. Red – False donor splice sites. Green - overlapping scores. Sequences were retrieved from the Altextron database (Clark and Thanaraj 2002).

5.5.3. Building a RNAbind model

Start the application: *RNAbind*

1. Select a Name for your model. Optionally add a short description.
2. Select an Alignment file containing a set of aligned sequences containing your signal.
3. Enter a Binding sequence to which putative signals will be bound to.
4. Optionally change default binding energies.
5. Select an energy Threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
6. Select a Colour for the graphical representation of hits of this model in **Seqool**.

Evaluate the model using the following options:

Show parameters

Display model parameters

Score sequences

Score a set of sequences. Use this option to compare score distributions of real signals and random or false sequences, for evaluating score thresholds, and for testing a model. Note that the sequences should have the same length as those of the alignment file used for training the model. For longer sequences additional nucleotides are ignored. The dialog offers some additional options, such as the saving of sequences above a score threshold.

5.6. Scoring the relative position of a signal: DistM

Module *DistM* 

5.6.1. Introduction

In some cases the binding efficiency of a signal depends on its location relative to another signal. E.g. the efficiency splicing enhancer signals or the polypyrimidine tract depend on the distance relative to the acceptor splice site. *DistM* allows to estimate the distance of a signal by scoring all subsequences within a given window size and finding the position with the highest score, which probably represents the putative signal. For example, for including the information of how far the polypyrimidine tract is located from the acceptor splice site, a *DistM* model can be used as illustrated in Figure 50.

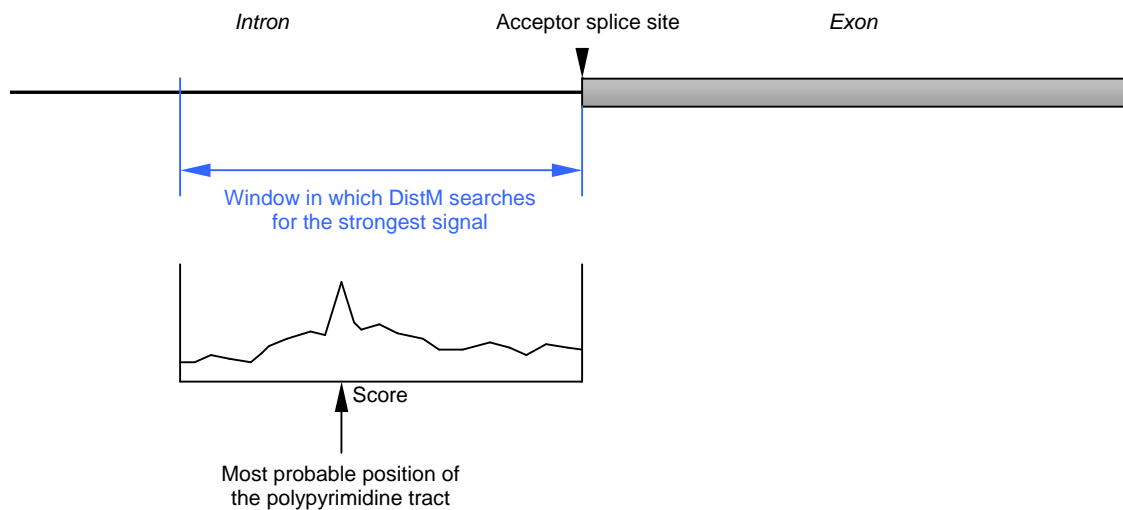


Figure 50. Scheme of the application of *DistM*: Within a given window size *DistM* searches for the strongest signal (here of the polypyrimidine tract) and returns the distance of that signal relative to the current position of the model (here the acceptor splice site).

For using a *DistM* model, a basic signal recognition model must first be provided for searching a signal within the indicated region (window). *DistM* will report the position of the best hit within this window (i.e. the position of the highest-scoring signal), not the score itself. Consequently, *DistM* can not be used for searching signals, but it is rather used as a submodel of other higher models, such as decision trees or neural networks.

5.6.2. Building a *DistM* model

Start the component: *DistM*, when starting up, make sure you choose the model directory containing your model files. The path containing any submodels to be used with *DistM* must be indicated before building the model. The model path can be changed by pressing the “Directory” button.

1. Select a Name for your model. Optionally add a short description.
2. Select the signal search Model which shall be used for searching the highest-scoring signal.
3. Select the Range in which to search with the specified model.
4. Press the Build button.


Evaluate the model using the following options:

Score sequences

Score a set of sequences. Use this option to compare score distributions of real signals and random or false sequences, and for evaluating score thresholds. The corresponding dialog offers some options, such as the saving of sequences above a score threshold.

6. Combining models

6.1. Hybrid Models: Adding or subtracting scores of several models

Module *HyM* 

The combination of two models may be useful for increasing the overall recognition performance. A simple way to combine models is to add or subtract the scores of each model. This will be demonstrated by the following simple example. The aim of this example model is to recognize human constitutive acceptor splice sites, or in other words, to distinguish human constitutive acceptor splice sites from “false” splice sites (subsequences containing AG but not being real splice sites). In a first step a PSSM using information content is constructed for the region -10..+2 nt of the acceptor splice sites. A score threshold of 4.95 results in 19.7 percent false positives (false splice sites recognized mistakenly as true splice sites) and 19.0 false negatives (true splice sites classified as false splice sites). Since coding exons usually have a higher codon usage, the recognition of splice sites might be enhanced by including this information to the model. This information can be added by combining the previous PSSM (-10..+2 nt) with two additional OFM models scoring codon usage upstream and downstream of the splice site (24 nt length each). Both models are combined with the previous PSSM using the component *HybridModels*. For combining the models each model file is loaded (button *Add*) and the position where to apply each submodel is specified (option *Search at position*): In order to analyse a set of true and false splice sites which contain the nucleotides +24 to +24 of the splice sites, or false splice sites respectively, the position of the PSSM is set to +10 (from the start of the sequence, i.e. 24-10). The OFM for codon usage is used twice, first at the beginning of the exon (position +24 nt, since the exon begins at position 24 in the sequences), and second at the end of the intron (position 0, since the length of the OFM is exactly 24 nt). Two OFMs are included, because codon usage is expected to differ between exons and introns (i.e. downstream and upstream the splice site). Since codon usage is expected to be high in the downstream exon, the score of the respective OFM is added to the score of the PSSM. The codon usage of the upstream intron, on the other hand, is expected to be low. Consequently, the score of this OFM is subtracted. The final settings of this hybrid model are shown in Figure 51. Score distributions for real and false human acceptor splice sites are shown in Figure 52.

An additional option (*Search for the best hit between*) allows to score the best hit for a signal only. This might be useful when the strength of a signal shall be measured, which possibly influences another signal. In this case the exact position of or distance between the signals

might not be essential for their recognition, rather it might be sufficient that both signals are observed near each other.

	Submodel	Search	Sc.	
0	Acceptor splice site, human, IC	+14	+	
1	Codon usage, 24 nt	+24	+	
2	Codon usage, 24 nt	0	-	
3				
4				
5				

Figure 51. Settings for a hybrid model, comprizing one PSSM for the human acceptor splice site, and two composition models for codon usage (see text for details).

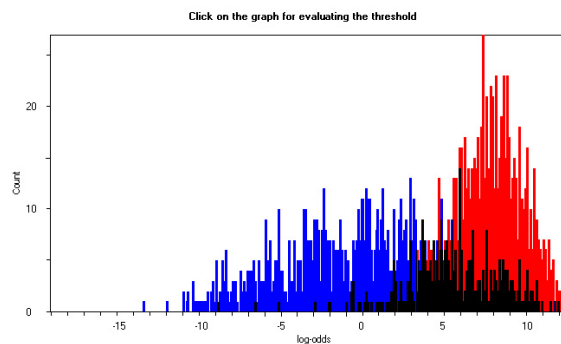


Figure 52. Score distribution of constitutive acceptor splice sites and false splice sites (intronic sequences near the splice sites containing AG, but not being confirmed real splice sites) calculated using a hybrid model combining a PSSM (-10..+2, using information content), an OFM using codon usage for the exon start (+1..24), and an OFM using codon usage for the intron end (-24..-1). The scores of the first two models are added, the score of the last is subtracted.

6.1.1. Building a hybrid model

Start the component: *HyM*, when starting up, make sure you choose the model directory containing your model files.


1. Select a Name for your model. Optionally add a short description.
2. Add models you want to combine.
5. Specify the Position where to apply the model. E.g. use position "0" for a model for the core acceptor splice site (covering e.g. bases $-10..+2$) and a model for exon codon usage (covering e.g. the first 6 codons) at position "+10", i.e. exon start. Alternatively, you may choose to score only the best hit within a region, i.e. the position where the highest scoring hit was found.
6. Select if you want to Add or Subtract the model's score.
3. Select a Score threshold. If the score of a putative hit is above this value, it is designated to be a positive hit. This option is necessary for searching with your model in **Seqool** or **SeqoolM**.
4. Select a Colour for the graphical representation of hits of this model in **Seqool**.
7. Press the Build button.

Evaluate the model using the following options:

Score sequences

Score a set of sequences. Use this option to compare score distributions of real signals and random or false sequences, and for evaluating score thresholds. The corresponding dialog offers some options, such as the saving of sequences above a score threshold.

6.2. Combining models using decision trees

Module *DecisionTree* 

6.2.1. Introduction

A decision tree provides a classification method which, in signal recognition, makes a series of observations about a putative signal and then classifies the signal by making binary decisions (yes/no) for each observation. Graphically, decision trees are trees with bifurcations, each bifurcation represents one decision. Figure 53 shows a simple decision tree for the recognition of constitutive human acceptor splice sites, as an example. Acceptor splice sites are measured by three models (results from these models are the “observations”), a PSSM for the positions – 10 to +2, and by two composition models, one for upstream codon usage (-24..-1) and another for downstream codon usage (+1..+24). These three models represent the nodes of the tree. For analysing splice sites, the model of each node analyses a putative signal sequence. Then, a classification decision is made based on the resulting score of the respective model and the process is repeated for the next node, until a terminal branch, representing the final classification decision, is reached. In this example, the decision tree first analyses the PSSM score. If the score of this model is below zero, then the sequence is most probably not a splice site (see to the score distribution of this PSSM, Figure 46). So the sequence will be classified as a *False splice site*. Otherwise, the sequence is analysed regarding the downstream codon usage, which is expected to be above 0 for real splice sites. If codon usage is below 0, then the sequence is probably not a real splice site, consequently it is classified a *False splice site*. Otherwise, upstream codon (which should be low for real constitutive splice sites) usage is analysed too. If upstream codon usage is high, then the sequence is classified again as a *False splice site*. Otherwise, the sequence is finally classified as a real *Acceptor splice site*.

With the present score thresholds, classification performance is still insufficient, because threshold have only been roughly guessed. Only 18.8 percent of real splice sites and 98.9 percent of false splice sites are recognized correctly. However, the module *DecisionTree* offers an automatic optimisation of thresholds. The optimised thresholds improve recognition to 86 percent for both real splice sites and false splice sites (14 percent false positives and 14 percent false negatives). Note that this decision tree performs better than the hybrid model mentioned in the previous chapter which used the same submodels.

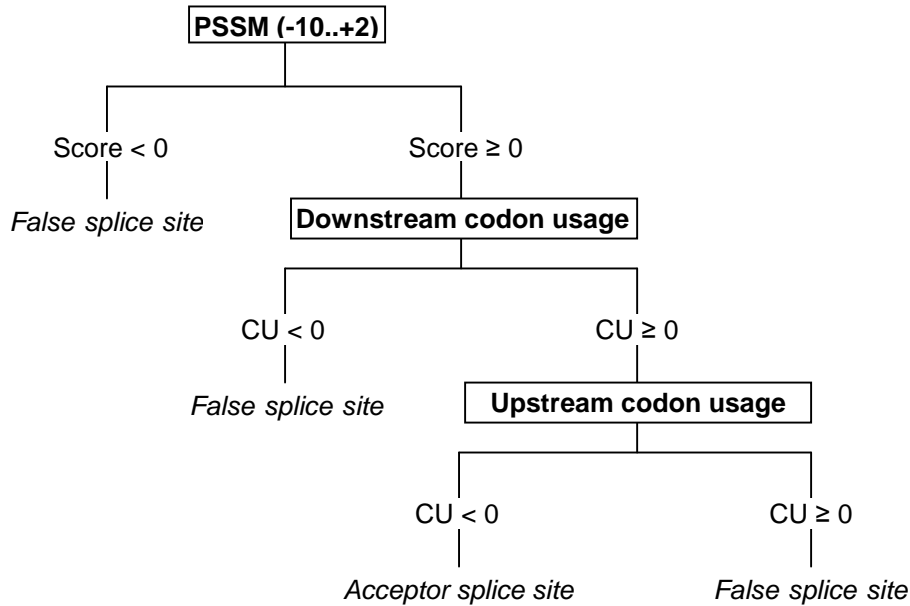


Figure 53. Example of a simple decision tree for the human acceptor splice site (constitutive).

6.2.2. Construction of a decision tree

In this example, the decision tree for human acceptor splice sites mentioned above will be constructed as an example of how to use the component *DecisionTree*. After starting the module *DecisionTree*, the tree is named and possibly a short description of the tree is provided. This example is based on the example file 'Acceptor splice site_human.det', which is available in the directory 'Examples\DecisionTree'. However, this file will not be loaded at this point, because it shall be demonstrated how to build a new decision tree from scratch.

Adding the first node

The construction starts with the addition of the first submodel, representing the first node in the tree. This node is labelled *Node 0* in the program. A submodel is added by pressing the *Add*-button. In the next windows which opens, a table located at the top of the window lists the nodes and models which are included in the tree so far (Figure 54). At present, no submodels (nodes) have been added. Below that table the model for the first node (*Node 0*) can be chosen by clicking on the *Select*-button. Now, a new dialog opens, which lists all models in the 'models' directory (this is the directory which is used by default, if this directory does not exist, then the program asks the user to select the directory containing the submodels at startup. However, it can be changed by pressing the *Directory*-button). Since the models for this example are located in the directory "Examples\DecisionTree" the directory has to be changed

to this path. Then, the model 'Acceptor splice site, human, IC (PSSM/WMM)' is selected from the updated model-list and the selection is confirmed by pressing the Select-button. Finally this node is named "Acceptor site PSSM".

In the next step, the position where to apply this first submodel has to be defined. This decision tree shall analyse putative splice acceptor sites from the -24 to +24 nt. The PSSM which was added for the first node only covers the range -10 to +2 nt. Consequently the search-position has to be set to +14 (the nucleotide -24 corresponds to the search position 0, the nucleotide +24 corresponds to the search position 48).

Now the two decisions have to be defined for the node. The decision will depend on submodel's score. If the score is below a certain threshold (i.e. the threshold set in the submodel itself), then one decision will be made, otherwise the other one. The aim of this first submodel is to exclude sequences which are clearly no splice sites and to pass those sequences which might be to another submodel. For sequences which are not real splice sites the PSSM will produce a low score. The decision for these sequences will consequently be to classify them as "false" splice sites. This is done in the program by selecting the option *Report hit* for in the section *Action when score < threshold*. Then the Text 'False splice sites' is added (optionally a colour can be chosen for displaying hits in the **Seqool** main program). Now all sequences which have scores below the score threshold of the PSSM, will be labelled 'False splice sites'.

The next step is to define how to treat sequences with a score higher or equal to the score threshold of the PSSM. For such sequences codon usage will be analysed by another submodel. This submodel will correspond to the next node in the tree, i.e. *Node 1*. Consequently, the option *Go to node* is selected in the *Action when score \geq threshold* box, and the *Node index* is set to '1' (although the respective node has not yet been defined). This completes the settings for the first submodel (Node 0). Figure 54 shows all settings for this node. Press *OK* for confirming all settings.

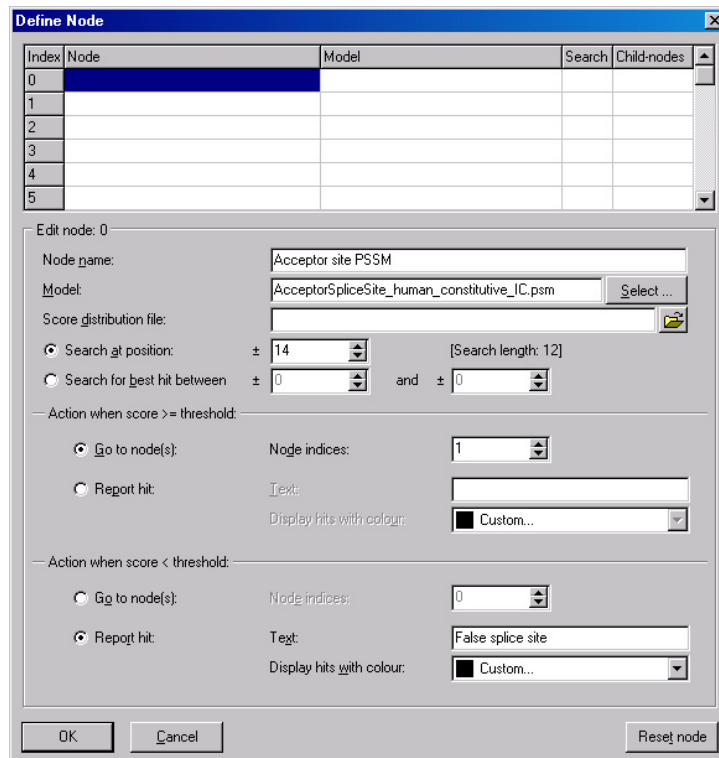


Figure 54. Settings for the addition of the first node in the decision tree.

Adding more nodes

The next node introduces a model for analysing downstream codon usage. The respective model (a codon usage composition model with the length 24 nt) is also available in the directory '\Examples\DecisionTree'. In order to add that model select the second row in the node list (in the main window) by clicking into the row and then press the *Add*-Button. As before, the node is named first, this node will be named 'Downstream codon usage'. The model 'Downstream codon usage, 24 nt (OFM)' is added and the search-position is set to +24, since the position is referring to the -24..+24 region of the splice sites. Since, codon usage in exons is expected to be high, sequences with low codon usage are labelled 'False splice sites'. This is done by activating *Report hit* in the bottom box *Action when score < threshold* (note that the threshold of the codon usage model is 0). If codon usage is high, then upstream codon usage will be analysed too. Thus the *Action when score \geq threshold* is set to the next node (*Node 3*). Finally all settings are confirmed by pressing *OK* and the last node is added. For this last node, the model 'Upstream codon usage, 24 nt (OFM)' is used and the node is labelled 'Upstream codon usage'. For high scores, i.e. high codon usage, false splice sites will reported (set *Report hit* and add the text 'False splice site'), because true splice sites should show a low upstream codon usage. Correspondingly, true acceptor splice sites are reported for low codon usage (set *Report hit* and add the text 'Acceptor splice site'). Finally all nodes of this decision tree have been defined. For this example the colour 'green' will be selected. This completes the basic construction of the decision tree.

If the decision tree will be used for searching in the main programs **Seqool** (or **SeqoolM**), then additional settings should be made. First, an additional option allows to choose a colour for displaying hits (in this case sequences classified as 'Acceptor splice sites') in the program **Seqool**. Additionally, the relative position where a hit should be reported by **Seqool** or **SeqoolM** has to be selected. The present model ranges from -24 to +24. Consequently, the splice site is located exactly in the middle, i.e. at +24 nt. In order to mark a hit exactly at the splice site (i.e. after the AG dinucleotide) set the option *Mark hit at* to +24. For visualizing hits in **Seqool**, the range where to paint hits can also be specified. If the red bar indicating a hit should range from the last two intron bases to the first two bases of the exon, then the range where to paint a hit is set to -2..+2 (options *Paint hit from ... to ...*). In this example only the intron AG should be marked, thus the range is set to -2..-1.

Completing the tree

Finally the decision tree is complete. Before applying the tree press the button *Build decision tree*, which causes the program to check all submodel files and node connections. After that, the option *Show decision tree* can be used to display a simple graphical view of the tree including the submodel files and decisions. The tree may be tested by pressing the button *Score sequences*. Note that sequences for testing the tree must correspond to the length of the tree (in this example they must range from -24 to +24 from the acceptor splice site).

6.2.3. Optimisation of thresholds

With the present score thresholds used in the submodels classification of the decision tree is still insufficient. Remember, that the score thresholds of all submodels were simply set to 0. These thresholds can be tuned manually in order to enhance the classification performance. However, it is time saving to use the automatic optimisation of thresholds which is provided by the component *DecisionTree*. During optimisation procedure thresholds of all (or selected submodels) are varied by random until the classification performance of the tree improves (this method is not as fast as the gradient descent method, but it is not that prone to get stuck in local minima).

For optimising thresholds press the button *Optimise thresholds* and select the submodels (nodes) whose threshold should be optimised (by clicking on the check boxes in the submodel list). Select two files, one containing real signals, and the other one containing false signals (e.g. random sequences). In the case of the example decision tree for the human acceptor splice site one file is provided which contains real splice sites (file 'AcceptorSpliceSite_human_constitutive_5000seqs_-24_24_LearningSet.msa') and another which contains false splice

sites (file 'FalseAcceptorSpliceSite_human_5000seqs_-24_24_LearningSet.msa'). For each file a decision must be assigned, i.e. either the decision 'Acceptor splice site' or the decision 'False splice site'. Since the first file contains a list of real splice sites the decision 'Acceptor splice site' is assigned to the first file. Correspondingly, the decision 'False splice site' is assigned to the second file.

Finally it has to be specified how to do the optimisation procedure (see options at the bottom of the optimisation dialog). The option *Precision* indicates the step with of random changes. For example, when a value of 2.0 is selected, then the score thresholds of the submodels are changed in steps of 2.0 (e.g. 2.0, 4.0, 0.0, or 6.0). The maximum deviation from the present threshold defines the maximum variation of the thresholds. For example, if the initial threshold is 0, the maximum deviation is 10, and the precision is 5, then the random thresholds tested are either -10, -15, -5, 0, 5, 10, 15, or 20. The *Number of runs* defines how many random changes are performed in total. If only a single node is optimised a few runs are sufficient (one run includes one random threshold assignment and the subsequent classification of the positive and negative sequence files). However for optimising thresholds of several submodels at the same time many runs are needed. After some time, when acceptable thresholds have been determined, the precision and the range of deviation can be decreased gradually, in order to fine tune the thresholds. The option *Decrease precision and deviation by 50%* defines how many runs to perform this.

Two more optimisation settings indicate what exactly should be optimised. The *maximization of the mean of correct decisions* maximizes the overall recognition. However, the recognition for the positive and negative sequence set might be very different. E.g. the mean correct recognition might be 90 percent. But sequences from the positive set might be recognized in 100 percent of all cases while only 80 percent of the sequences of the negative are recognized correctly. In many cases it is preferable to maximize the mean performance, but to minimize the difference of correct decisions between both datasets at the same time too.

When running the optimisation (press button *OK*) the submodel's thresholds are iteratively changed and all sequences in both sequences sets are classified. The user is informed about the improvement of the classification during the whole process. At the end (or when the process is stopped by the user) the user is asked to confirm if the new thresholds should be saved in the submodel's files. Evidently, scores should be saved only if the scores result in an improved classification performance. If the thresholds are far from being adequate, never save the score thresholds, because otherwise no further optimisation might be possible: E.g. if the optimal score threshold of a given model amounts to 0.0, then saving the model with a

threshold of -20.0 is detrimental, because in the next optimisations the score 0 may never be reached again, depending on the selected range of variation.

Finally it should be noted why two separate submodels for codon usage were used in this example. The reason is the optimisation of the thresholds. The use of two separate files allows to save two different (optimised) score thresholds in the files. If a single was used, then only a single threshold could not be optimised separately.

6.3. Combining models using neural networks

Module *BPN*et 

6.3.1. Introduction

Neural networks are machine learning methods which are inspired by neurobiology. A neural network contains several “neurons”. A neuron receives information from other neurons and passes information to other neurons. However, in contrast to real neurons, the neurons in a neural network are rather simplified. The connection between neurons is regulated by weights. It is the adjustment of these weights, which allows a neural network to “learn” to recognize some kind of pattern. Figure 55 shows a scheme of an artificial neuron.

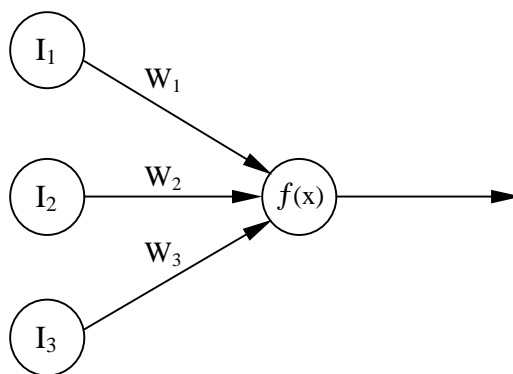


Figure 55. Scheme of an artificial neuron. Input neurons (I_1 - I_3) pass a signal to the neuron. The strength of the signal from each neuron is adjusted by a weight (W_1 - W_3). The incoming signals are summed up and an activation function ($f(x)$) determines the output of the neuron.

The incoming signals are summed up and evaluated by an activation function (the program uses a sigmoid activation function, i.e. $f(x)=1/(1+e^{-x})$), which determines the final output of the neuron.

A large number of different network architectures and learning methods exist. The module BPN*et* uses a backpropagation network, which is a rather universal and robust network type. Backpropagation networks consist at least of three layers of neurons, one input layer, at least one hidden layer, and one output layer (Figure 56). The input layer comprises neurons which simply pass information to the hidden layer without modification. The hidden layer can be interpreted as the basic part of the neural network, containing some kind of multidimensional representation of all information. The final output layer shows the results of the network. A putative signal (or pattern) is classified (or recognized) by the neural network depending on

which output-neuron is activated most. Each output-neuron represents a classification decision, e.g. one neuron may represent the decision 'the analysed pattern is a restriction site for EcoR1', and the second one may represent the decision 'the analysed pattern is not a restriction site for EcoR1'. If the output-neuron representing the decision 'the analysed pattern is not a restriction site for EcoR1' is activated most, then the neural network indicates that no restriction site was recognized.

The number of nodes in the input layer depends on the number of signals which shall be analysed by the network. The number of output nodes depends on the number of classes which shall be predicted. If the outcome of a prediction is binary (e.g. is a splice site or is not a restriction site), then two nodes are sufficient. For recognizing hand-written letters 26 output nodes would be needed. The number of neurons in the hidden layer may be arbitrary. It depends on the specific task of the network how many nodes to use (and how many hidden layers to use), which can be done experimentally by adding or deleting hidden neurons and observing the performance of the network.

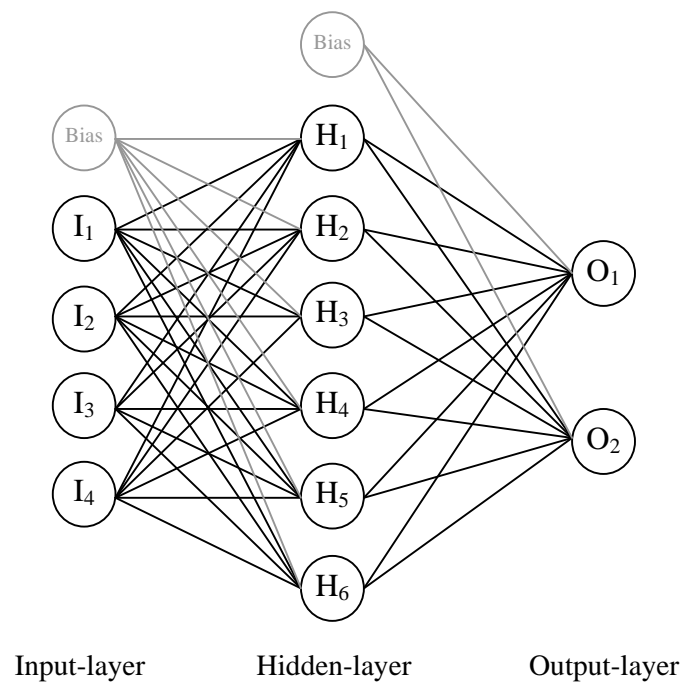


Figure 56. Scheme of a backpropagation network with three layers. For details see text.

Two different kinds of parameters could theoretically be adjusted during the training of a backpropagation network, the weights and the threshold of the activation functions. In order to adjust only the weights, a bias neuron is added to all layers, except the output layer, so that the weights from the bias neurons adjust the threshold of the activation function.

6.3.2. Training of backpropagation networks

The training process of backpropagation networks generally includes the following steps: 1. Random initialization of all weights. 2. Presentation of data for which the classification is known. 3. Calculation of the resulting output values of each node (beginning with the input nodes and ending with the output nodes, *forward pass*). 3. Comparison of the output of the output-layer with the true classification. 4. Calculation of the mean square error for each node and adjustment of the weights connecting to a node using a learning rule and beginning with the output nodes and ending with the input nodes (*backpropagation*). 5. Repetition of all previous steps until the mean square error is minimal.

The adjustment of each weight during training is achieved by gradient descent, i.e. by changing a weight by a small value (learning rate) in the direction, in which the error decreases. Since error minimization may get stuck in local minima, a momentum is usually added, which "pushes" the weight over small local minima in the current "learning direction" (standard momentum algorithm). The respective learning rule including this momentum term is: $\Delta w_{ji}^q(t+1) = \eta \cdot \delta_i^q \cdot O_j^{q-1} + \alpha \cdot \Delta w_{ji}^q(t)$, where Δw_{ji}^q is the weight change of the i th weight of node j in layer q , δ_i^q is the respective error, η is the learning rate, O_j^{q-1} is the output activation of the previous layer's neuron, and α is the momentum (ranging between 0 and 1). For a more detailed description of neural networks see e.g. Rumelhart et al. 1986. There are also excellent introductions to neural networks available on the internet.

6.3.3. Over-learning

Neural networks are prone to over-learning (see also chapter 6.4). Over-learning occurs when a neural network adapts too much to the sequences in the training set and loses its ability to abstract, which results in a good classification performance of the known sequences in the training set, but to an insufficient classification of unfamiliar new sequences. Over-learning can be tracked by using two different sets of sequences, one for training, and another one exclusively to observe if over-learning is indicated (cross-verification set). *BPNet* allows to split a set of sequences automatically into two sets, one for training and one cross-verification. During the training process, the classification error (mean square error) should be observed in both sets. When the error does not decrease for the cross-verification set any further (or if it starts to increase again), the training process should be stopped. At this point the neural network starts to lose its ability of abstraction.

6.3.4. Example of a neural network for the human acceptor splice site

In following a simple example of a backpropagation network will be build for human constitutive acceptor splice sites. The model will analyse the splice site from -24 to +24 nt. The same three submodels will be used as in the previous example of a decision tree, i.e. a PSSM using information content for the nucleotides -10 to +2 of the splice site, a composition model measuring codon usage in the last 24 nt of the upstream intron, and a composition model measuring codon usage in the first 24 nt of the downstream exon. These models correspond to the input neurons of the network. The output layer of the model will consist of the two output neurons 'Acceptor splice site' and 'False splice site'. The respective example model file 'Acceptor splice site_human.bpn' can be found in the directory 'Examples\BPNet'.

After starting the module *BPNet*, a name and a short description for this model is provided. The first model, the PSSM, is added by pressing the Add-button. In the following dialog the directory has first to be changed to the example directory 'Examples\BPNet' which contains all submodel files used in this example. The directory is selected by pressing the Directory-button. Note that all submodel files used for the network have to be located in one and the same directory! The first submodel is added by selecting the model 'Acceptor splice site, human, IC (PSSM/WMM)' in the box *Available submodels*. For each submodel it has to be defined at which position (relative to the complete model, i.e. the neural network) it has to be applied. Since the backpropagation network will focus on the region -24..+24 nt and the PSSM covers the nucleotides from -10 to +2, the search position 14 is entered (option *Search at position*), which is the position from the start (the nucleotide -24 corresponds to the search position 0, the nucleotide +24 corresponds to the search position 48). Finally settings are accepted by pressing *OK*. Now the next model is added by pressing the Add-button again. This time the model 'Upstream codon usage, 24 nt (OFM)' is selected (option *Available submodels*) and the search position 0 is entered (option *Search at position*), since this model shall cover the nucleotide -24 to -1. Again the settings are accepted by pressing *OK*. Finally the last model is added by pressing the Add-button again. This time the model 'Downstream codon usage, 24 nt (OFM)' is selected and the search position 24 is entered, since this model shall cover the nucleotide +1 to +24. After accepting all settings by pressing *OK* again, all submodels are added.

Now the settings for displaying hits should be entered. Since the model focuses on the nucleotides -24 to +24, the actual splice site will be located at the position 24. Therefore, the option *Mark hit at* is set to +24. Graphically, hits shall correspond to the AG dinucleotide of the acceptor splice sites. Consequently, the positions for which a hit shall be displayed are the

nucleotides -2 and -1. Enter these numbers in the respective edit fields (options *Paint hit from...to*). The current settings are displayed in Figure 58.

BPNet models can be pre-processed by other model types, e.g. position specific score matrices (PSSM). Pre-processing models are used to exclude sequences from further analysis by the network, e.g. such sequences which are evidently not target sequences. Therefore, the use of a pre-processing model increases the speed when scanning many sequences with a *BPNet* model, and it may also be used to increase the accuracy of classification. In the present example a simple pre-processing model will be used in order to exclude all sequences which do not contain the AG-dinucleotide at the central position of the splice site from further analysis. This model (a simple PSSM for the nucleotides A and G) is provided in the example directory 'Examples\BPNet' (file: 'Preprocessing model_AG.psm'). To add the pre-processing model press the button *Select model* in the pre-processing section. Then choose the model 'AG (PSSM/WMM)' and set the search position to the +22, so that the pre-processing model covers the AG-dinucleotide of the core splice site (see Figure 57). Finally, confirm the settings by pressing *OK*. Now the model is ready for training. The final settings are shown in Figure 58.

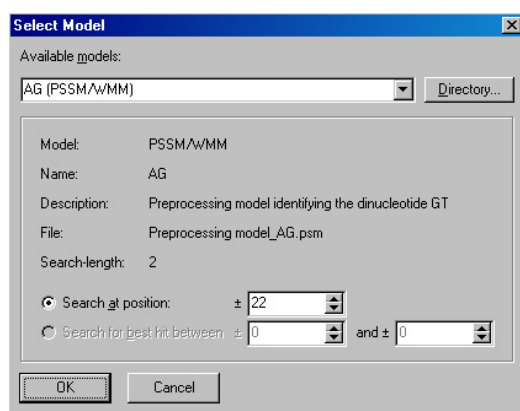


Figure 57. Selection of the pre-processing model.

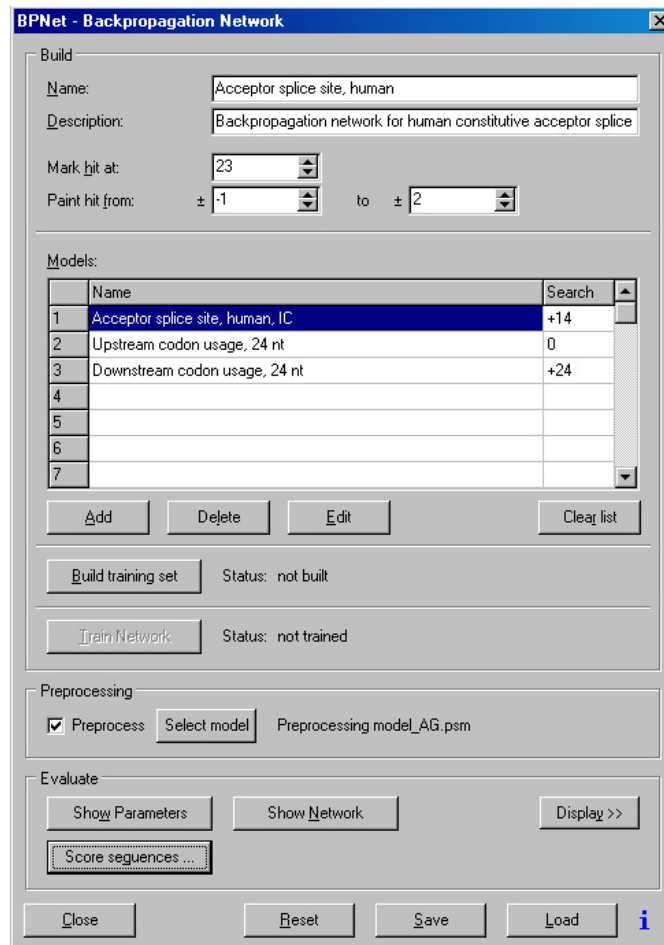


Figure 58. Settings of a backpropagation model for human acceptor splice sites after the addition of the submodels.

For training, two sets of sequences are needed, one set containing real acceptor splice sites (file 'AcceptorSpliceSite_human_constitutive_5000seqs_-24_24_LearningSet.msa') and another one containing false splice sites, i.e. sequences containing AG, but not being real splice sites (file 'FalseAcceptorSpliceSite_human_5000seqs_-24_24_LearningSet.msa'). For optimal classification all training set should be of equal size, i.e. each should contain the same number of sequences. Both sets will later be subdivided in order to create a training set and set for cross-verification. By pressing the *Build training set* button, a dialog opens which allows to choose all files to be used for training. Files can be added by clicking in the respective cell of the table (column *File*). First the file 'AcceptorSpliceSite_human_constitutive_5000seqs_-24_24_LearningSet.msa' is chosen by clicking in the first row of the table (column *File*). This file contains the confirmed, real splice sites, therefore this file is assigned to the class 'Acceptor splice site' (enter the text 'Acceptor splice site' in the second column of the table). Next, the file 'FalseAcceptorSpliceSite_human_5000seqs_-24_24_LearningSet.msa' is added by clicking in the second row of the table (column *File*). The assigned class is 'False splice site'. The current settings for building the training set are shown in Figure 59. The option *Exclude sequences with*

a *bad score* should be activated. This excludes all sequences, which contain invalid letters (other letters than those allowed for the current molecule, e.g. for DNA only A, G C, and T are allowed) or which are too short. Since scores can not be calculated for these sequences, the respective submodels report a score of -100. The option *Display scores* shows each score of each submodel and for each sequence.

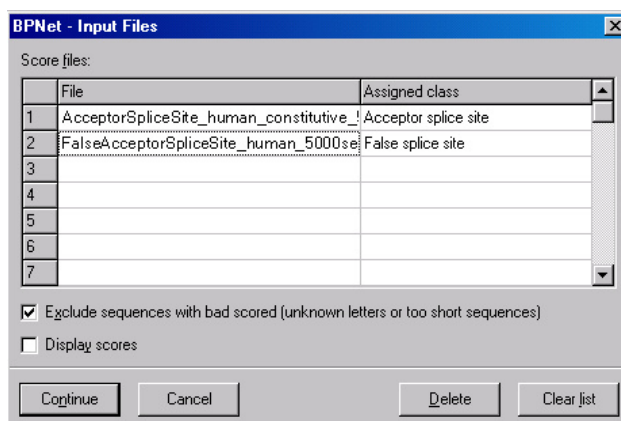


Figure 59. Settings of the dialog for building the training set.

After pressing *Continue*, another dialog appears for adding additional settings which are only relevant when the network is used in the main programs **Seqool** or **SeqoolM**. This dialog allows to select the colour for the graphical display of hits and which class not to report. For this example, only real splice sites should be reported. Therefore, the option *Don't report hits for class* is set to the class 'False splice site'. For the displaying hits of real splice sites the colour red is selected by clicking in the respective row of the table. Figure 60 shows the settings for this dialog.

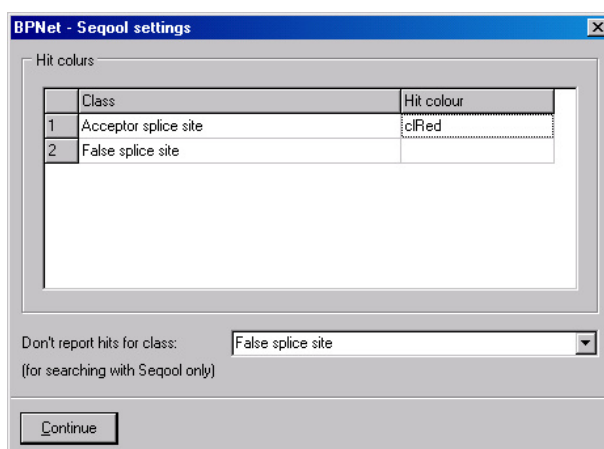


Figure 60. Settings for the display of hits for the main programs **Seqool** and **SeqoolM**.

After pressing *Continue*, scores are calculated for all submodels and all sequences which are provided in the two given files of real and false acceptor sites. Subsequently, a final dialog appears, which prompts the user to split the total set into a training set and a set for cross-verification of given size (Figure 61). This dialog proposes a subdivision into two equally sized halves. However, the user is free to change the size of the subsets or even to select not to use a cross-verification set. After pressing *Continue*, the sets are built and saved into a temporary file.

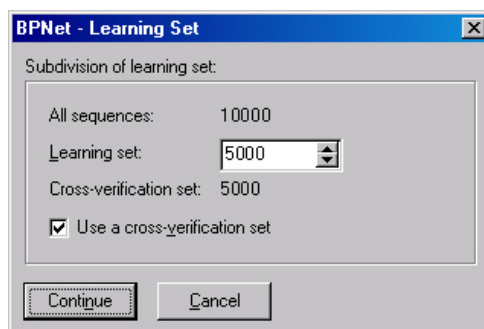


Figure 61. Dialog for the subdivision into a training set and a set for cross-verification.

The network is trained by pressing the *Train network* button, which opens a new window with a variety of options. In the top of the window, a graph shows the change of the error (mean square error) of the network during the training process. This error will be shown for the training set and the cross-verification set. On the right side of that graph, some basic information is listed, i.e. the number of training process performed (epoch), the mean square error (MSE) of the training and the cross-verification set, and the performance of the network for both subsets (training and cross-verification).

Settings: Network architecture

The lower boxes allow to make settings for the network and the training process. The network architecture can be changed by setting the number of layers (three layers, i.e. one input, one hidden, and one output layer, are usually sufficient). The number of input neurons corresponds to the number of submodels used. The best number of hidden neurons (Hidden units layer 1 or layer 2) should be determined experimentally. The number of output neurons is defined by the number of classes used. In this case two classes ('False splice site' and 'Acceptor splice site') were defined.

Settings: Training settings

The learning rate (η) determines the speed of the learning process. A high value increases the speed, but decreases accuracy. The momentum is the value α (see introduction). The default

value will usually perform well. The option *randomise weights before start* has to be activated for the initial training process. If the initial training was insufficient and the network has to be trained further, then this option can be deactivated. Batch- and Online-training refers to the training method. For online training, weights are adjusted after each single sequence of the training set. For batch training, weights are adjusted only after scoring all sequences. This results generally in a higher accuracy, but it slows the training process down.

For each sequence each output neuron will have an output activation ranging from 0 to 1. For example the output neuron for the class 'Acceptor splice site' and the class 'False splice site' might amount to 0.90 and 0.03, respectively. The neuron which shows the strongest activation shows the most probable class of the given sequence. In this case the activation is strongest for the class 'Acceptor splice site' which indicates that the sequence is probably a real splice site. The high activation of this node compared to the low activation of the other node shows that the classification is quite reliable, whereas activations of 0.60 and 0.48 would indicate an unreliable classification. The accept threshold determines at which output activation to accept a classification of a sequence. This allows to exclude classifications which are not very reliable. In the present example all sequences will be classified, so the accept threshold is set to 0.5.

Settings: Stop training

The training process can either be stopped after a given epoch, in dependence of the mean square error, or if the mean square errors of the training and the cross-verification set indicate over-learning. An alternative way for avoiding over-learning is to estimate the epoch when over-learning begins by trial and error.

Display options

During the training process the changing weights of the neural network can be shown graphically. Options for customizing the display include *Display connection weights*, *Actualize after each epoch*, and *Display models and classes*.

Training

For this example a neural network with one hidden layer and 10 hidden neurons will be used. The training method batch-training is selected, a learning rate of 0.5 is selected, and the training will be stopped after 2000 epochs. For the remaining training settings default values will be used. The training is started by pressing the Train-button. Figure 62 shows the decrease of the mean square error of the neural network during the first 1000 epochs.

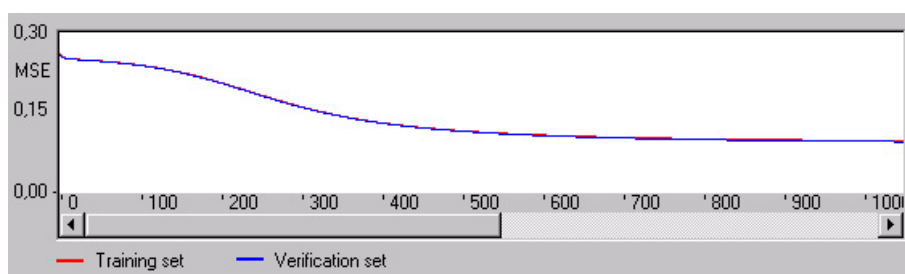


Figure 62. Mean square error (MSE) decrease during the training process of the backpropagation network (x-axis: number of epochs).

After training for 2000 epochs, the final classification performance was 87.9 percent for the training set and 87.4 percent for the cross-verification set. Over-learning was not indicated, since the mean square error of the cross-verification set continuously decreased. Figure 63 shows the architecture of the trained network. Notably, the strongest weights (absolute values) in the input layer are found for the PSSM submodel and the model for downstream codon usage. This indicates that these models are more relevant for splice site recognition than the model for upstream codon usage.

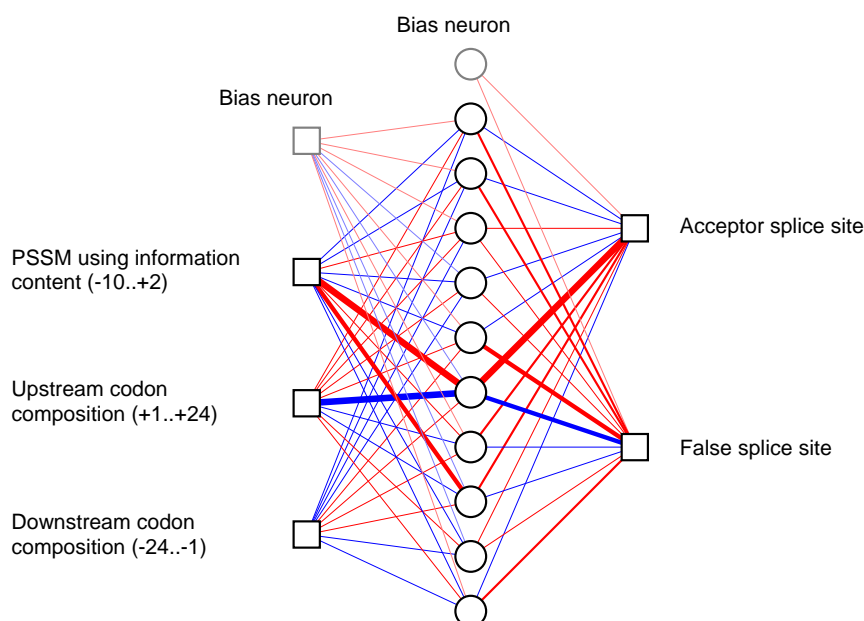


Figure 63. Architecture of the trained backpropagation network for constitutive human acceptor splice sites. Red connections between neurons indicate positive weights, blue connections indicate negative weights. The absolute value of weights is reflected the thickness of the connections.

More details about the classification performance can be displayed by pressing the Classify-button. This information includes the number of false and true positives and negatives, and the relation of correctly identified (more correctly classified) sequences in relation to the total

number of sequences $C/(C+W)$. It results that 91.1 percent of real acceptor sites and 84.7 percent of false acceptor splice sites is classified correctly for the cross-verification set (see Figure 64), which is a notable increase compared to the classification performance of the decision tree example presented earlier. However, the performance could still be increased by including more information of the sequences, e.g. hexamer frequencies (as in the default BPNNet acceptor splice site model provided with **Seqool**). The sensitivity and specificity of the example model can easily be obtained using the results in the classification dialog (Figure 64, verification set!).

Class	Trainings set					Verification set				
	Correct	Wrong	Unknown	C/(C+W)	Performance	Correct	Wrong	Unknown	C/(C+W)	Performance
1 Acceptor splice site	2248	252	6	0,899200	0,897047	2266	219	2	0,911871	0,911138
2 False splice site	2114	364	5	0,853107	0,851389	2120	380	3	0,848000	0,846984
3										
4										
5										
6										
7										

Figure 64. Classification performance of the neural network for the training set and the cross-verification set.

The sensitivity is the fraction of correctly identified signals, i.e. the number of true positives (TP) relative to the number of all true signals (true positives and false negatives, FN, together). For the present model 2266 true positives (correctly identified splice sites, see Figure 64) and 219 false negatives (not recognized real splice sites) were observed. Additionally, 2 sequences could not be classified (NC). Then the sensitivity amounts to $TP / (TP+FN+NC) = 2266 / (2266+219+2) = 0.911$, or 91.1 percent. The specificity is the fraction of true negatives relative to the number of all false signals. In the present example, 2120 true negatives were identified, while 380 (and 3 unclassified sequences) were not correctly recognized. Therefore the specificity amounts to $TN / (FP+TN+NC) = 2120 / (380+2120+3) = 0.847$ or 84.7 percent. It becomes clear, that the performance values listed in the classification-dialog are in fact the sensitivity and the specificity. In the dialog, these values are only labelled performance, because one might use many more classes for classification, not just one “true” (here for the real splice sites) and one “false” class (here for the false splice sites).

6.4. Over-learning

Over-learning refers to the process that a model extracts too much information from the training set during training. It then enhances the recognition of the sequences in the training set, but it loses the ability of abstraction, which leads to a decreased recognition of sequences which are not part of the training set. Over-learning must be dealt with, especially in neural networks, but also in decision trees, where the automatic optimisation of thresholds represents a learning process. Over-learning can be detected by the use of cross-verification set during training. In the PSSMs, MDDs, or OFMs, pseudocounts (or providing expected frequencies for words which are not included in the training set) help to prevent that a model sticks too much to the sequences it was built on.

One effective method for preventing over-learning is to divide all the sequences into a training set, used for the actual training of the model, and a cross-verification set, used to detect over-learning. At the beginning of the training process the recognition performance should increase for both sets. After a certain point, the recognition performance will still decrease for the training set, but increase for the cross-verification set. This indicates over-learning and the training process should be stopped at this point (Figure 65).

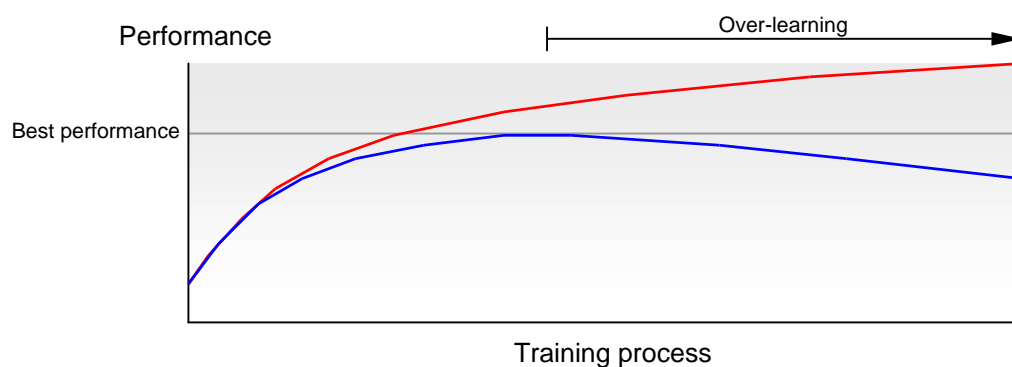


Figure 65. Scheme of the learning process for a training set (red line) and set for cross verification (blue line). After a beginning increase of performance for both datasets, the performance starts to decrease for the cross-verification set, while it still increases for the training set. This indicates over-learning.

7. *FastAFormat* - formatting sequences and much more

FastAFormat allows various functions for extraction and manipulation, and formatting of sequence or text files. For selecting an input file press *Open infile*. The selected file is displayed in the upper text field (for very large files, it might be useful to deactivate the option *Preview infile*, which prevents files being displayed). A number of options are accessible on different program pages. Select any options in a given sheet and press *Process sequence*. In the opening file dialog select the location where the processed file is saved to.

7.1. Functions

Tab-sheet *Format FastA*: Formatting FastA files

<i>Get subsequences</i>	Get a subsequence of a given range
<i>Filter sequences</i>	Filter sequences which are shorter or longer than a given size
<i>Read sequences from end</i>	Create reverse sequences (this is not equal to creating a homologous strand)
<i>Revert sequences to homologous strand</i>	Creates the homologous strand of a sequence (only for DNA)
<i>Convert sequence to</i>	Converts the sequence letters to lower or upper case
<i>Break lines longer than</i>	Creates line breaks after a given number of chars
<i>Add sequence number</i>	Adds the number of each sequence in after the '>' character

Tab-sheet *Get FastA*: Creating FastA files

This page is intended for extracting sequences from unconventional file formats. The use is illustrated by the following example:

```
ID: 1859562Hg6
Sequence: aatgctagcttcgcgatcaxgcgacgtgacagtttccccaag
          ggctcgatcggatcgatgctagctagctgatcgatcgtagcta
          gcgcatcgctatcgtagctagctagctgatcgatcgtagct
          gcgcatctctatcgtagtctactatctctactagcatgatc
          atctgatcgatcgtagctagcgcgataacgatatacaacta
```

<i>Description</i>	Enter the text indicating that a line contains a description or ID. (In the example enter 'ID:')
<i>Add '>' character</i>	Add '>' to the description if necessary (In the example, the line containing the ID does not include the '>' character necessary for FastA-files, therefore this option has to be activated)
<i>Add line number</i>	optionally add the number of each sequence to the description
<i>Get sequences starting with text</i>	Enter the text indicating the begin of a sequence (in the example enter 'Sequence:')
<i>Cut letters on the left/right of lines</i>	Optionally adjust how many letters have to be cut on each side of a line, if a sequence does not cover the whole line (in the example 9 letters have to cut on the left of each line)
<i>Preview</i>	Press preview to test if sequences are extracted correctly

Tab-sheet *Format text*

Range and length

<i>Get columns</i>	Extract a given column from a text file. Columns can be recognized automatically when indicated by tabs or spaces, other wise it has to be indicated which character defines a column.
<i>Truncate lines</i>	Cut a given number of characters from the left or right of each line
<i>Get substring</i>	Extract a substring from each line
a) <i>Number of chars to copy</i>	Extract a given number of characters from one side of a line
b) <i>Save strings containing</i>	Extracts a subsequence from each line if a line contains a given substring. The subsequences can be extracted including neighbouring characters (options "from" and "to").
c) <i>Save substring from...to...</i>	Extract a subsequence from within a line

Additional filters

<i>Save lines if/if not</i>	Save lines only under certain conditions
a) <i>Character found at position...</i>	Save/do not save lines if a given character is found at a given position
b) <i>Column equals...</i>	Save/do not save lines if a given text is found in a given column
c) <i>Line contains</i>	Save/do not save lines if lines contain a given substring
<i>Delete empty lines</i>	Do not save empty lines or lines containing only spaces

Tab-sheet *Misc. text*

<i>Replace with</i>	Replace any text with another text (optionally case sensitive)
<i>Convert to</i>	Convert to upper /lower case
<i>Save no. of lines</i>	Save a given number of lines only (from the begin)
<i>Fill lines with character until length is...</i>	Fill the end of each line with a given character until the length of the line equals a given number

Tab-sheet *Get random lines*

Get a given number of lines from a file. Lines are selected by random.

8. Distribution and download of models from the *Seqool* website

Seqool users are encouraged to publish their own models to the scientific community if they are of scientific significance. The *Seqool* website provides an interface for registered users for uploading their models, and for downloading models published by other users (Figure 66). For model download login at <http://www.biossc.de/seqool/download.html> and proceed to the download area. Then click on a model name for download and copy the model file in your model directory (by default this is the directory “/models”, however you might have changed that). Models which contain several submodels (decision trees, backpropagation networks, hybrid models) can be downloaded as zip-files. For using these models extract the respective files into the model directory. Note that any file downloaded from the internet and consequently also models uploaded by other users may possibly be infected with viruses. Although users may only upload *Seqool* models and zip-files, it is recommended to scan recently downloaded files, especially zip-files for viruses before use.

Model download

Model	Description	Author	Institution	Reference
Acceptor splice site, human	Model for constitutive human acceptor splice sites, using information content, the model covers -15..+2 nt relative to the splice site	Magnus Wang	Seqool	-

[Access models provided by Magnus Wang](#)

Model upload

Note: Models will be listed together with the name and institution of the author.

Model name (binding factor/signal)

Model description (please include also the model type, e.g. PSSM)

Reference (if the model was described or referred to in a scientific publication)

Select model file

If a model includes several submodels, e.g. a decision tree or neural network, provide a zip/rar archive containing all model files. Uploaded files must not exceed 200 kb.

Show my e-mail to other users.

Figure 66. The download area of the *Seqool* website provides an interface for down- and uploading models which can be used with the *Seqool* program package.

For uploading a model provide a model name (e.g. the name of the binding factor) and a description of the model, which should include the model type (e.g. PSSM), the length of model, and other relevant information. If the model has been referred to in a published article, you may also include the reference to that article. Finally select the model file to upload the model. Simple models which contain only a single file (WMM, PSSM, MDD, profile HMM,

RNABind, OFM) may be uploaded directly. Models which include submodels (such as decision trees, backpropagation networks, or hybrid models) must first be compressed to a single zip-file. This zip-file must include the main model file and all necessary submodel files. The zip-file may then be uploaded.

Models provided by Magnus Wang [Hide]					
Model	Description	Author	Institution	Reference	
Acceptor splice site, human	Model for constitutive human acceptor splice sites, using information content, the model covers -15..+2 nt relative to the splice site	Magnus Wang	Seqool	-	Delete

Figure 67. Users may access and delete their own models after clicking on “Access models provided by...”.

When preparing a model which contains many submodels for upload, it is recommended to modify the model file extensions. Instead of using “xxxxx.xxx” use “xxxxx.xxx_”. E.g. instead of using “model.psm” use “model.psm_”. This will prevent that submodels are visible in the programs **Seqool** or **SeqoolM**. For example, a decision tree may contain many submodels. These submodels might never be used alone, i.e. outside the decision tree. But since they are located within the model directory (together with the main model, the decision tree) they also appear in the model lists of **Seqool** and **SeqoolM**, enlarging that list unnecessarily. However, if the submodels are named e.g. “model.psm_” instead of “model.psm” then they disappear from the model lists of **Seqool** and **SeqoolM**.

9. References

- Banerjee H., Rahn A., Davis W. and Singh R. 2003. Sex lethal and U2 small nuclear ribonucleoprotein auxiliary factor (U2AF⁶⁵) recognize polypyrimidine tracts using multiple modes of binding. *RNA* 9, 88-99.
- Burge C. and Karlin S. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, 268, 78-94.
- Burset M. and Guigo R. 1996. Evaluation of gene structure prediction programs. *Genomics* 34: 353-367.
- Clark F. and Thanaraj T.A. 2002. Categorization and characterization of transcript confirmed constitutively and alternatively spliced introns and exons from human. *Hum. Mol. Gen.* 11, 451-464.
- Durbin R., Eddy S., Krogh A., and Mitchison G. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge – New York.
- Gribkov M., Devereux J., and Burgess R.B. 1984. The codon preference plot: graphic analysis of protein coding sequences and prediction of gene expression. *Nucleic Acids Res.* 12: 539-549.
- Rumelhart D.E., Hinton G.E., and Williams R.J. 1986. Learning, internal representation by error propagation. In: Rumelhart D.E. and McClelland J.L (eds.). *Parallel Distributed Processing: Explorations in the microstructures of cognition*, Vol. 1., MIT Press, Cambridge, MA.
- Schneider T.D. 1997. Information content of individual genetic sequences. *J. Theor. Biol.*, 189, 427-441.
- Schneider T.D., Stormo G.D., Gold L. and Ehrenfeuch A. 1986. The information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188, 415-431.
- Senapathy P., Shapiro M.B., and Harris N.L. 1990. Splice junctions, branch point sites, and exons: sequence statistics, identifications, and applications to Genome Project. *Methods Enzymol.* 183: 252–278.

10. Appendix

10.1. File formats used in Seqool

Alignment files (plain text) are used for building models, such as profiles. They include raw sequences:

```
gagtcctgtgttttgtgggtggcaggtggggagacagaagaggagaaga
ggtgacagctgttttctgcctcaggagaaactgaagccagaatacttg
ccacacattcttggccttctgcagatcacctttagatttctcgcgc
ctcgcggcgctgtgctgcgcgcaggtggcggtaaggctggaaaggac
ctcgcggcgctgtgctgcgcgcaggtggcggtaaggctggaaaggac
gtttgtgtatgcttaaaatttaagttcccagtgggccgtattcatcga
caactgacagattctgccttttaggtacttgaactggcaggaatgca
ccggccctcttctctgtccccagctcagcaacagcacgacggctggc
ttgattgccctcctcccactgcagatccattacaccggctgctctatg
cagaactcttt...
```

Frequency files (and background frequency files, *.fre) contain frequencies of nucleotides or oligonucleotides. (oligo-)nucleotides. The following file contains the typical DNA base frequencies of human exons:

```
0.243001    ← Adenine
0.27215     ← Cytosine
0.279091    ← Guanine
0.205758    ← Thymine
```

Frequency files list only the frequencies and not the (oligo-) nucleotides to which the frequencies refer¹. However, frequencies are listed in the following order: A, C, G, T. Oligonucleotide frequencies are listed analogously, e.g. for dinucleotides the order is: AA, AC, AG, AT, CA, CC, CG, CT.

Word reference set files (*.ref) are used to extract over- or under-represented words (e.g. oligonucleotides) from a set of sequences in *SeqoolM*. They contain a list of all words found in a reference set, their respective frequency, and the percentage of sequences in which the word was found.

¹ Frequency files do not include the names of (oligo-) nucleotides because they are meant to be applied in *Seqool* applications only; for creating a table including (oligo-) nucleotides and the respective frequencies simply use the function `Statistics|Oligonucleotides` (see chapter 4.3.3).

```
cga      <- word
0.004    <- frequency of word
0.2      <- percentage of sequences in which word was found
acg
0.00443103
0.21
cgt
0.00446552
0.217
...
```

10.2. Performance of example models

Mode name	Model type	Sensitivity	Specificity	TP, FN, TN, FP
Acceptor splice site, human, IC (-15..+2)	PSSM / WMM	0.909	0.905	90.9 %, 9.1 %, 90.5 %, 9.5 %
Acceptor splice site, human (-36..+24)	Neural network	0.942	0.914	94.2 %, 5.8 %, 91.4 %, 8.6 %
Donor splice site, human (-15..+10)	MDD	0.928	0.925	92.8 %, 7.2 %, 92.5 %, 7.5 %
Donor splice site, human (-15..+10)	Neural network	0.950	0.922	95.0 %, 5.0 %, 92.2 %, 7.8 %

Table 6. Performance of example models for the recognition of constitutive human splice sites provided with *Seqool*.