INTERNSHIP REPORT

Adequate Null Models for Profile Hidden Markov Models for Protein Families

prepared for the
Salzburg University of Applied Sciences
Degree Program
Information Technology & Systems Management

submitted by:
Franz Felix Auer

Head of Faculty:	FH-Prof. DI Dr. Gerhard Jöchtl
Supervisor:	FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl

Salzburg, February 2009

Details

Franz Felix, Auer:	Franz Felix Auer
University:	Salzburg University of Applied Sciences and Bowling Green State University
Degree Program:	Information Technology & Systems Management
Title of Internship:	Adequate Null Models for Profile Hidden Markov Models for Protein Families
FH-Salzburg Supervisor:	FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl
BGSU Supervisor:	Dr. Larry O. Hatch

Keywords

Profile-HMM, Bioinformatics, Protein Families, Null Model, Sequence Alignment

Abstract

Profile Hidden Markov Models are a common approach to search for similarities among proteins. The scores produced by such HMMs need to be normalised by so called null models. This paper is a report on the search for and implementation of more sophisticated null models for the protein scanning software of the University of Salzburg.

Contents

List of Figures

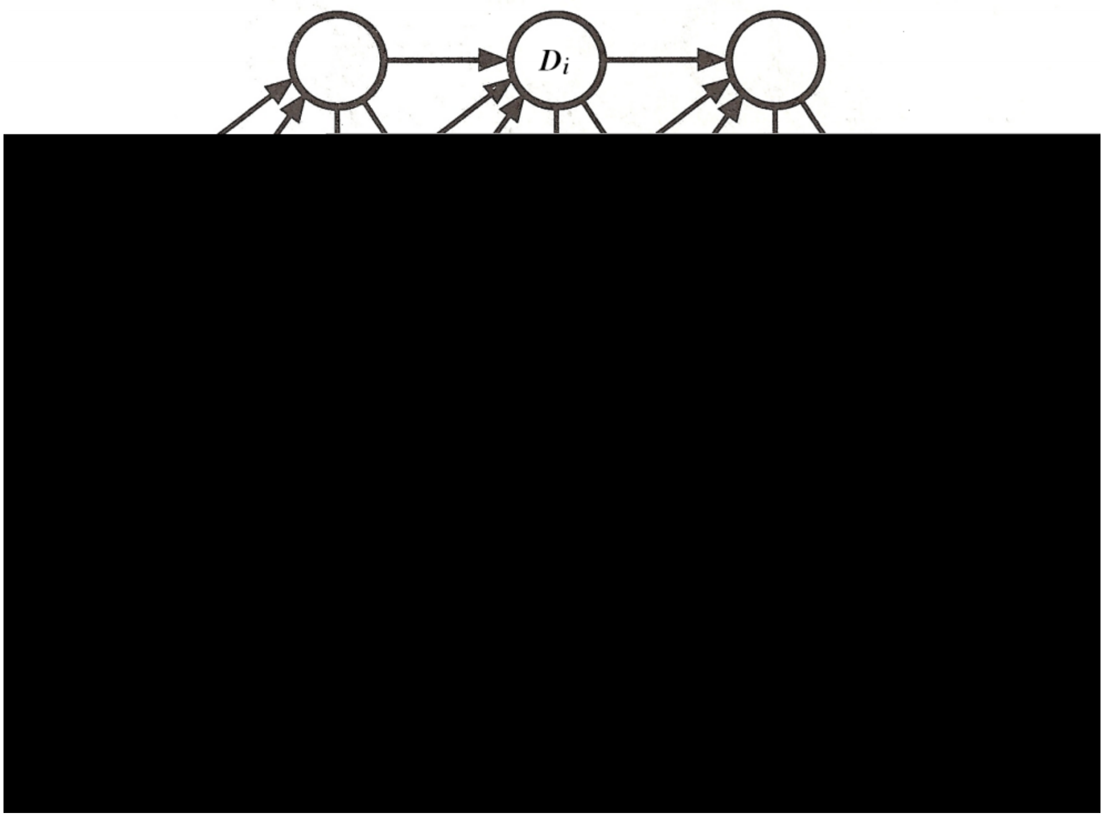
List of Tables

1 Introduction

Despite the fact this was a bioinformatics project, there is not much required to know about biology in order to understand this piece of work. The basic scope was to improve the performance of an existing software solution. This software has been designed to recognise proteins from a database that fit the category searched for. These proteins are represented by chains of characters which by itself represent the 20 different amino acids of which proteins consist of. Originally, proteins are three-dimensional constructs, yet, for processing them, they are being simplyfied to these one-dimensional chains. Hence, every protein is now represented by a sequence of characters. To actually process such sequences, there are different approaches.

1.1 Profile-HMM

The approach the software follows is the so called Profile-HMM. Fundamentally, a Hidden Markov Model (HMM) is a set of different states that either do not emit characters, so called silent states, or do emit characters with a certain character distribution, so called non-silent states, interconnected by transition probabilities. The purpose of such an HMM is to either emit a certain sequence of characters, or to calculate the probability of a given sequence. In this case, only calculating the probability of a given sequence is of interest. Figure 1.1 shows the so called Plan-9 Profile-HMM used in this software. The elements of such a model are match states, represented by squares, insert states, represented by diamonds, and delete states, represented by circles, as well as



[redacted]

Figure 1.1: Plan9-Profile-Hidden Markov Model[2]

arrows that represent the state transitions and these transitions' probabilities. While the delete states are silent states, the insert states emit characters from a background distribution, which is the same for every insert state. Every match state has its own trained distribution from which it emits characters.

A profile-HMM is characterised by its abilty to be trained from a certain set of parameters. That set defines the background probabilities for the insert states, the distributions for the match states, the number of match states, and all the transition probabilities. That training process is executed with a Multiple Sequence Alignment (MSA), which basically is a set of protein sequences of the same family. Further information on that process is given at [2, 6].

The Plan-9-HMM can be subdivided into three parts: The begin cycle, at the lower left side, the main cycle at the upper side, and the end cycle at the lower right side. Both the begin and the end cycle consist of a looping insert state and a delete state with transitions to, respectively from, all match states. These two cycles are used to come by the fact that most sequences are much longer than the part of it that is actually of interest for classifying it as a protein. Hence, the parts of the sequence before and after that part are simulated by these cycles. As already mentioned, the number of match states is variable, thus, the number of columns in the match cycle is variable, too. An image with five columns was only chosen for better illustration[2].

1.2 Null Models

The same reason that coins the need for begin and end cycles is one reason for the need for null models: The highly variable sequence length. As the transition probabilities are always below one, the HMM's score for a sequence lessens with the sequence's length. To make these scores comparable, a null model is needed. A null model is another, usually a lot simpler, HMM that the sequence is scored with as well. The score from the model is divided by that score, which ideally nullifies the length dependancy.

Apart from that basic requirement, it is desired that a null model also cancels out high scores that arise from sequences that do not fit the family's structure but only have a similar composition of amino acids. More detailed explanations can be fund at [1].

Another aspect of null models, that is of particular interest for this software suite, is their quality in points of superfamily and fold distinction. However, the outcome is strongly related to the quality of the trained model, hence, the impact of the null models to that distinction is rather small. Proteins are seperated into three categories of similarity, family, superfamily and fold. While family represents the highest similarity, fold represents the least.

1.3 Coding

The software had been written in JAVA, hence any improvements had to be written in JAVA, too. Apart from that, Matlab was used to process and illustrate results. Only relevant fragments of the code written are shown in this paper. However, the complete code listing will be within the appendix of the follow up diploma thesis that will build on this piece of work.

2 Null Models

Before any improvements have been made, the software divided all raw results by the results of a simple null model. This simple null model is very similar to the begin and end cycle of the Plan-9-HMM. It is a single state HMM consisting of a begin state, a transition to a looping insert state and a transition from the insert state to an end state. All the transitions are the same as they are in the begin and end cycle, and the insert state holds the background distribution. Hence, the length dependancy of both the begin and the end cycle cancel out. However, the different transition and emission probabilities within the trained model leave a length dependancy. This dependancy is varying, as both the total number of model columns and the number of model columns the processed sequence actually passes differ. Furthermore, the simple null model does not take into account any knowledge about the distribution of the trained model and so does not lessen high scores that arise only from a similar distribution[1, 5].

Fortunately, there already are numerous approaches to solve these issues. The following sections deal with these approaches which are part of similar software suites, either Sequence Alignment and Modeling System (SAM) or HMMer. Further information on these suites can be found in the SAM Documentation [4] and the HMMer User Guide [3].

2.1 The former SAM Null Model

Before the reverse sequence null model was introduced, SAM solved the problem of length dependance and distribution bias by a complex null model that resembled the trained HMM. The only difference between those two models is that the emission probabilities of the match states in the null model are the same for all positions. Using such a null model solves the problem of length dependance, to compensate bias, the emission probabilities have to be dealt with. There are several approaches how to calculate these probabilities. The two papers [1, 5] entirely embrace these approaches.

2.1.1 Background Distribution

Using the background distribution of amino acids over all proteins is common in sequence analysis. This distribution only needs to be calculated once, which adds to speed and simplicity. However, it only corrects for bias in proteins, which is not sufficient at all. Furthermore, the application intended to improve here uses that very distribution, so there would be no improval in compensating bias.

2.1.2 Model Adaption

Another way to obtain an emission probability distribution for match states in the null model is to take a look at the model. During training, the model acquires the distribution bias of the training data set. If the distribution in the null model is related to that distribution, the bias can be canceled out. In the case of SAM, the geometric mean over the emission probabilities of the model's match states is built and modified to sum to unity. This mean distribution serves as distribution for all match states of the null model. HMMer also uses this approach for the distribution in its null2 model, yet with another method to fit the model.

2.1.3 Sequence Adaption

Instead of to the model, the distribution can also be adjusted to the scoring sequence. One example of this approch is setting the distribution in the null model's match states to the distribution of the sequence to be scored. Unfortunately, this method not only cancels out the compositional bias but in some cases even decreases the actual score too much.

2.1.4 Mathematics

Using a null model with the same complex structure as the scoring model is just double the calculating effort at first sight. Yet, with one preference met, it can be quite fast. If, within the null model, the emission tables in the match states as well as in the insert states are the same, the null model score can be calculated quite easily.

Usually, the log-odds score is just the logarithm of the model score divided by the null model score:

$$s = log \frac{P(D|M)}{P(D|N)} \ . \tag{2.1}$$

Both the scoring model and the null model are embraced by free insertion modules, which are identical and hence cancel out of the log-odds score. Therefore, only the score of the sequence's part inside the scoring model is relevant. As the emission probabilities throughout the null model are the same, the null model score can be seperated into two factors:

$$P(D|N_c) = P(\,|D|\,|N_c) * P(D|N_s) \ . \tag{2.2}$$

The complex null model score N_c is the product of the probability that a sequence D of the sequence length $|D|$ fits the complex model and the sequence score for the single state null model N_s. This calculation is much faster than calculating $P(D|N)$.

2.1.5 Annotations

None of the sources gives an idea how to calculate and interpret $P(\,|D|\,|N_c)$, hence, there is a need for further investigation. Furthermore, no mathematical proof for (2.2) is given. Also, the geometric null model coins some problems with families that have strictly conserved residues, if those are rare. If that is the case, false positives and inflated scores occur for sequences that have a compositional bias towards these rare residues. So, the requirement for balancing any compositional bias is not fully met. Ultimately, the need for the null model and all insertion modules having the same emission tables decreases the possibilities in adjustments to the HMM.

2.1.6 Outcome

2.1.6.1 Benefits

- composition bias canceled out for many families
- no length dependance
- fast

2.1.6.2 Drawbacks

- inflated scores for strict conservations of rare residues
- mathematical background unclear
- decreased flexibility

2.2 The Reverse Sequence Null Model

Since 1998, SAM has been using another approach for the length dependance and bias issue, the reverse sequence null model. The basic procedure is scoring the model with a null model which is the reverse trained HMM. As the reverse sequence has both the same length and the same composition, this approach fully eliminates those issues. The article [5] thoroughly deals with the concept of the reverse sequence null model.

2.2.1 Mathematics

Fortunately, the score of a reverse model is equivalent to using the score of the reverse sequence on the initial model. The reverse sequence null model not only meets the requirements, it also is very easy to implement.

$$s = log\frac{P(D|M)}{P(D_r|M)}\ ; \tag{2.3}$$

The score is just the log-odds ratio between the sequence score of the trained HMM $P(D|M)$ and that for the reverse sequence D_r. This is simple, yet very time consuming, as two fully independent HMM path calculations are required.

2.2.2 Annotations

The reverse sequence null model fullfils the requirements and appears to solve even more problems which were not in the focus of this research. However, this comes at the cost of almost double scoring time. SAM comes by this issue by introducing a threshold that determines at what score, calculated with a single state null model using the background distribution, the reverse sequence null model is triggered.

Concerning our application, the reverse sequence null model has another benefit. The idea of reduced alphabets does not fully vanish in an averaging process. Yet, whether it preserves its function or not has to be examined.

2.2.3 Outcome

2.2.3.1 Benefits

- simple implementation
- requirements fit
- reduced alphabet considered
- threshold as expert parameter

2.2.3.2 Drawbacks

- high processing time

2.3 HMMer Null Model Correction

Initially, HMMer calculates the log-odds score of a sequence using a single state null model. The emission probability table for this model is the general distribution of amino acids over all proteins. After that score is calculated, another table is calculated as the geometric mean of the a posteriori distribution of the actual sequence. This distribution is used in another single state null model. With the score computed with this model, a correction value is calculated. This value is used to correct eventual distribution bias in the sequence, in order to prevent false positives. The correction approach used by HMMer is capable of implicating numerous null models, yet HMMer uses only the two mentioned. All the information about the HMMer null model is taken from [3].

2.3.1 Derivation of the Correction Value

The basic scoring approach for wether a model fits a sequence or not, is the Bayesian probability theory, which concentrates itself on the question if a hypothesis H fits an observed data set D. This is calculated by dividing the product of the probabilities of the data fitting the hypothesis and the hypothesis itself by the product of the probabilities of the data fitting every possible hypothesis and the probabilities of every hypothesis:

$$P(H|D) = \frac{P(D|H)P(H)}{\sum_{H_i} P(D|H_i)P(H_i)} \ . \tag{2.4}$$

The following is an adjustment of (2.4) to the domain of HMMer. D represents the Sequence to be scored, while M is the model, standing for the Hypothesis to be tested and N is the null model. Here it is assumed that there are only two Hypotheses, the model and the null model:

$$P(M|D) = \frac{P(D|M)P(M)}{P(D|M)P(M) + P(D|N)P(N)} \tag{2.5}$$

HMMer now assumes that the model and the null model are equiprobable

$$P(M) = P(N)\ . \tag{2.6}$$

whiche simplifies (2.5) to

$$P(M|D) = \frac{P(D|M)}{P(D|M) + P(D|N)}\ . \tag{2.7}$$

Up to now, the log odds-score was being calculated for only one null model,

$$s = log\frac{P(D|M)}{P(D|N)}\ ; \tag{2.8}$$

which, combined with (2.7) adds up to

$$P(M|D) = \frac{e^s}{e^s + 1}\ . \tag{2.9}$$

Now, the log-odds score is being adjusted to multiple null models,

$$S = log\frac{P(S|M)P(M)}{\sum_i P(S|N_i)P(N_i)}\ . \tag{2.10}$$

so that the probability of the model fitting the sequence is now:

$$P(M|D) = \frac{e^S}{e^S + 1}\ . \tag{2.11}$$

For every other null model, a log-odds score, relative to the first null, is computed:

$$s_i = log\frac{P(D|N_i)}{P(D|N_1)}\ ; \quad for\ i > 1\ . \tag{2.12}$$

Furthermore, every other null model gets a likelihood relative to the first null model:

$$\Pi_i = log\frac{P(N_i)}{P(N_1)} \; ; \quad for\; i > 1 \; . \tag{2.13}$$

As (2.6) is not always true, a correction factor is necessary:

$$\Pi_M = log\frac{P(M)}{P(N_1)} \; . \tag{2.14}$$

All this sums up to a corrected score of

$$S = log\frac{e^{s_m+\Pi_M}}{1+\sum_{i>1} e^{s_i+\Pi_i}} \; , \tag{2.15}$$

which can be simplified to

$$S = (s_M + \Pi_M) - log\Big(1 + \sum_{i>1} e^{s_i+\Pi_i}\Big) \; . \tag{2.16}$$

This score is dependent of the two parameters (2.14) und (2.13), which need to be approximated. HMMer approximises them as $\Pi_M = 0$ and $\Pi_2 = \frac{1}{256}$ for $i = 2$. The distribution of HMMer's null2 model is calculated by averaging the distributions of the alignment's state path.

2.3.2 Outcome

2.3.2.1 Benefits:

- composition bias canceled out
- fast calculation owing to the single state null models
- easy implementation
- numerous null models possible
- two further expert parameters

2.3.2.2 Drawbacks:

- Exigency to calculate default values for the two parameters
- Dependance between the speed of the approach and the complexity of the null models used.

3 Application of the Null Models

Previously, the application has been using only a simple null model. Such a single state null model is also being used by HMMer and is part of its null2 model. The mathematical theory behind that null2 model promises both good results and high speed. Hence, the null2 model was chosen to be implemented and tested for matching the application's targets.

The second null model chosen to be tested was SAM's reverse sequence null model. According to the SAM manual, SAM does not use its old null model, spoken of in section 2.1 , anymore, but relies on the reverse sequence null model. Given as a reason for this is that the reverse model covers all the old model's advantages while adding some features. The only drawback of the reverse model is its significant need of processing time. Still, SAM's old null model is not of particular interest and only the reverse sequence null model has been implemented and tested[4, 3, 5].

3.1 Implementation

As already mentioned, only the HMMer null2 model and the reverse sequence model have been chosen for implementation. The implementation effort put into these two models highly varyied.

3.1.1 HMMer null2 Model

For the HMMer null2 model, the mathematics shown in chapter BLA!! have been realised step by step. With the knowledge of that chapter, the following code is self-explaining:

```
HMM.backTrack = result.getBackTrack() ;
null2Corr = this.null2Score(result.getBackTrack(), line, in) ;
null2Corr = null2Corr - HMM.simpscoreplain ;
null2Corr = null2Corr - Math.log( Pi2 ) ;
null2Corr = Math.log(1+Math.exp(null2Corr)) ;
HMM.null2score = HMM.simpscore + PiM - null2Corr ;
```

The function null2Score gives back the null2 score for the current sequence. That score is calculated with a single state null model carrying another character distribution table. This table is calculated as the geometric means of the distributions of the states in the trained model the sequence has passed. That path is represented by the string backtrack:

```
protected double null2Score(String[] backtrack,String sequence,
Viterbi in){
        double null2score = 0 ;
        int index = 0 ;
        backTrack = backtrack ;
        fillTable();
        for (int i=0;i<backtrack.length;i++){
                if (backtrack[i].matches("M.")){
                        addScoreM( i, in );
                        index++ ;
                }
                else if (backtrack[i].matches("I.")){
                        addScoreI( in ) ;
                        index++ ;
                }
        }
        for (int i=0;i<probTable.length;i++){
                probTable[i] = probTable[i] / index ;
        }
        null2score = scoreTable ( sequence , probTable ) ;
        return null2score ;
}
```

3.1.2 Reverse Sequence Model

As the reverse sequence model score is just the score of the reverse sequence passing the trained model, the implementation was very easy. The character sequence is just being reverse and scored again with the trained model.

3.2 Evaluation

To test the null models, a set of trained protein-family models has been scored against a database of known proteins. That database is an astral database, version 1.71.95, which consists of amino acid sequences and descriptions to every sequence, including the protein family. Within that database, the different families are labelled with a letter, followed by three numbers, seperated by dots. With that information, it is possible to attach every scored sequence to a category. These categories are family, super-family, fold and others, according to the sequence's relationship to the family the model has been trained with. Thus, it is possible to prognose how well every model would perform, if the information about the sequence's family was not given. Each model's performance was examined regarding the distinction between the four categories.

3.2.1 Characteristics

To measure the scoring performance of the different null models, several numbers have been calculated for each null model. The *medians*, *minima* and *maxima* of the scores of each category have been calculated for both every single trained model and for trained models altogether. To illustrate these numbers, the density functions for each category and null model have been plotted.

Additionally, the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) for different thresholds have been calculated, and the Receiver Operator Characteristic (ROC) has been plotted. These graphs display how well the different null models are able to distinct the different categories from lower categories.

3.2.2 Medians and Densityfunctions

All the data extracted from the test runs has been examined. The overall data, as well as the density graphs and numbers for particular training sets, picked after significance for the purpose of judging the null models, are given here.

Category	*N*ull Model Type		
	Simple	*Reverse*	*HMMer*
Family	60.633	67.323	60.486
Super-Family	-8.091	0.899	-8.167
Fold	-8.669	0.108	-8.703
Other	-8.993	0.049	-9.036

Table 3.1: Median values over all scores for all trained sets

Apart from a resemblance between the results of the simple null model and the hmmer null model, a significant offset between these two models and the reverse sequence null model can be identified from these numbers. Furthermore, it seems that all three of them classify the family values rather well, while classifying the super-family and fold values rather poorly. However, it is not possible to state anything about how well the null models perform in comparison to each other, as the differences between the category scores are almost the same for all three models. A look at the density function of the results of two family parameter files gives more information.

The focus for these density functions lies on the distinctability of super-family, fold and other values. The graph boundaries have been set to values that preserve readability by cutting off most of the family density function. The family values are so much higher that they would render the graphs hardly readable. Moreover, the family values are distinct so well that a density plot is not needed for them.

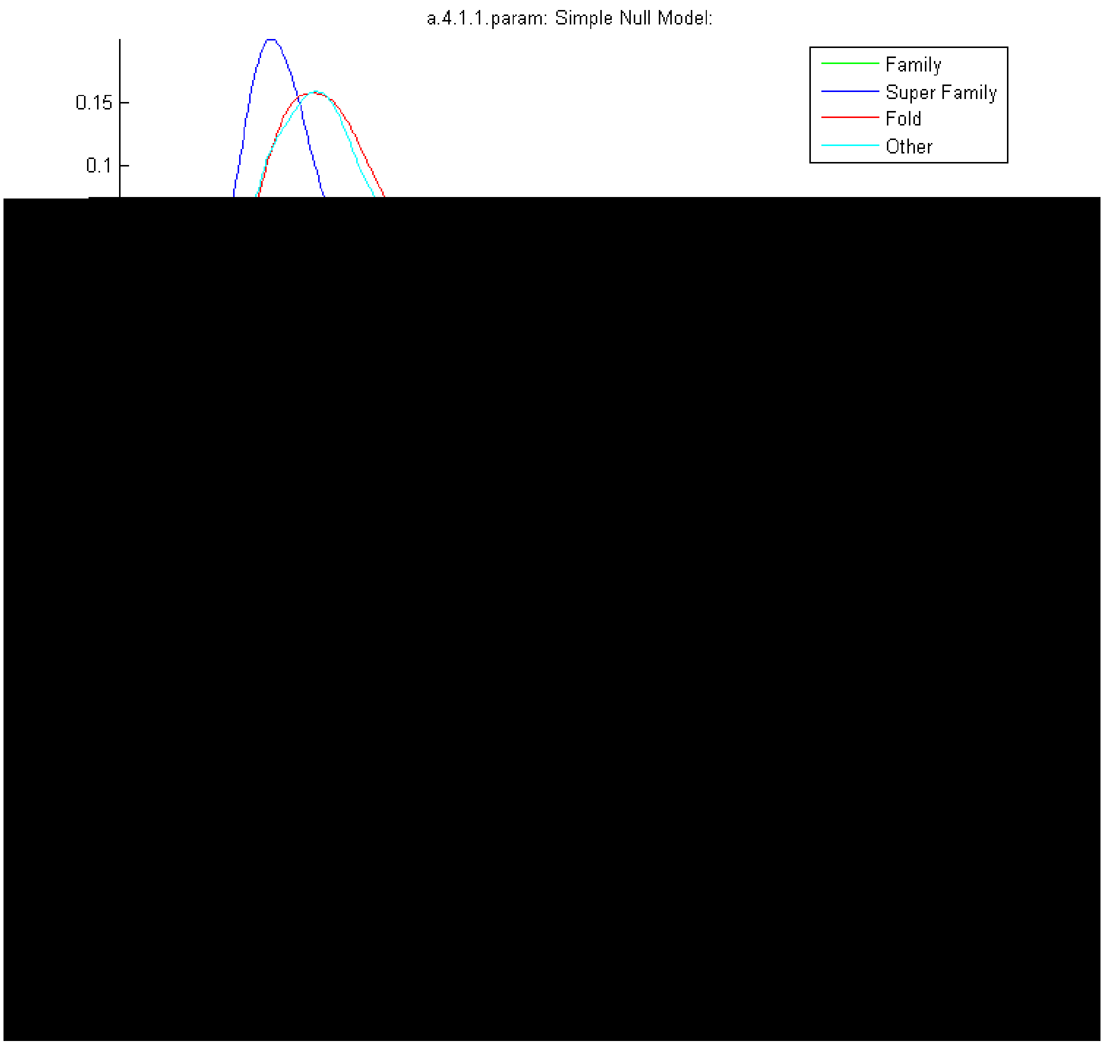


Figure 3.1: Density functions for family a.4.1.1

Unfortunately, only two of the parameter files create enough hits for super-family and fold to support an assumption like this density function. For the two examples, family a.4.1.1 and family a.4.5.28, shown in figures 3.1 and 3.2, the database provides 29 super-family and 95 fold hits, respectively 66 and 76.

However, apart from another indicator for the likeliness of the simple null model's and the HMMer null model's results, these graphs only show that none of the models makes it possible to properly distinct between super-family, fold and other values. All density graphs considered, only the family a.22.1.3, shown in figure 3.3, revealed a significant difference between the simple null model and the HMMer null model. In this case, the HMMer null model performs worse than the other two. However, that difference only shows up for the density curve of family values and there are only five family hits for

Figure 3.2: Density functions for family a.4.5.28

this family. The medians and maxima only differ by less than 0.01, yet the minima differ by over 22. Hence, not enough information is given from the density functions to tell quality differences between the three models.

Statistical classification methods like the ROC provide additional information and make statistically founded conclusions possible.

3.2.3 FRR, FAR and ROC

A more accurate and distinctive method of evaluating testsets is calculating the FRR, FAR and the ROC. The FRR gives the probability that a value that is classified as negative is in fact a positive one. Consequently, the FAR gives the probability that

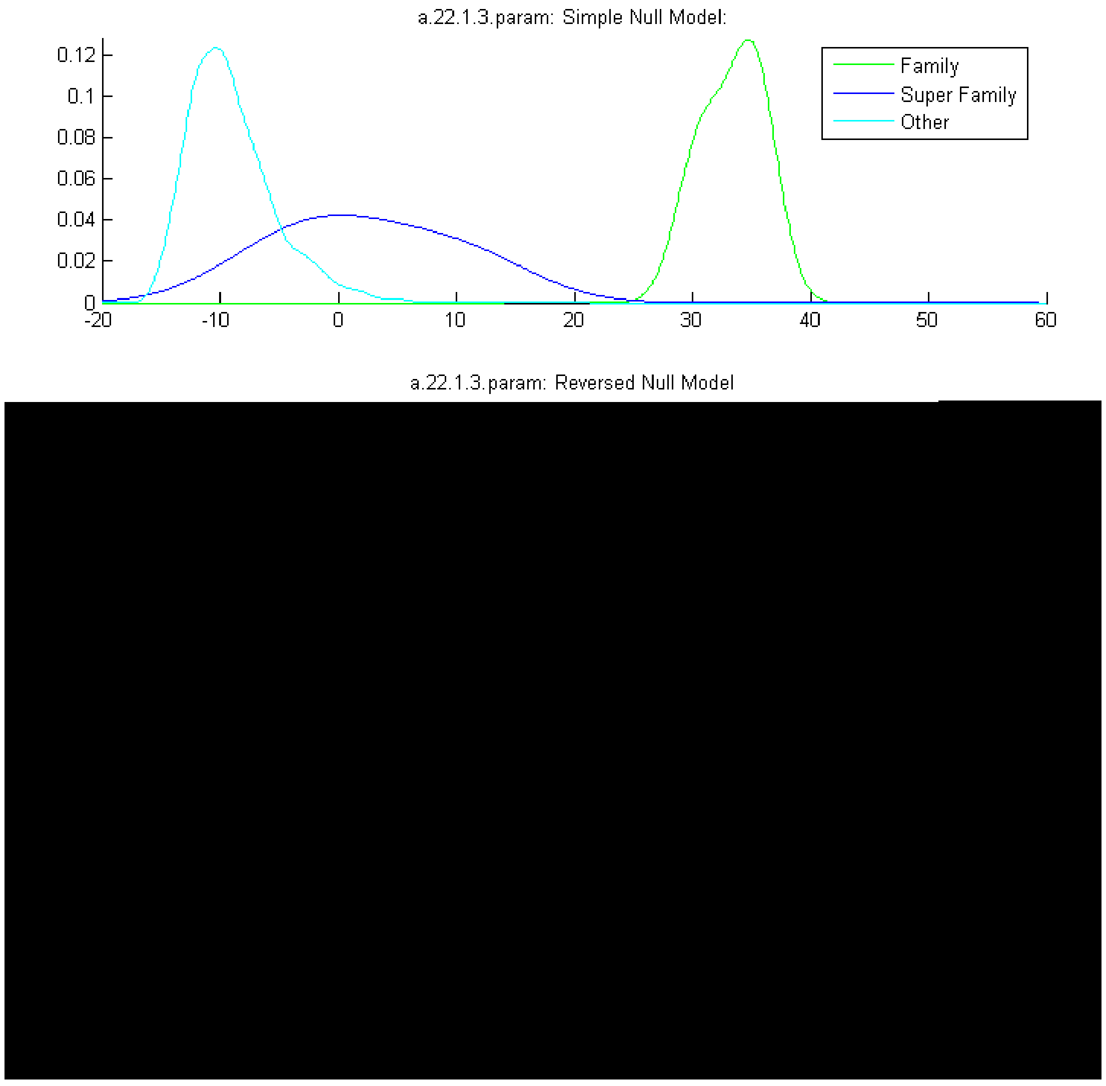
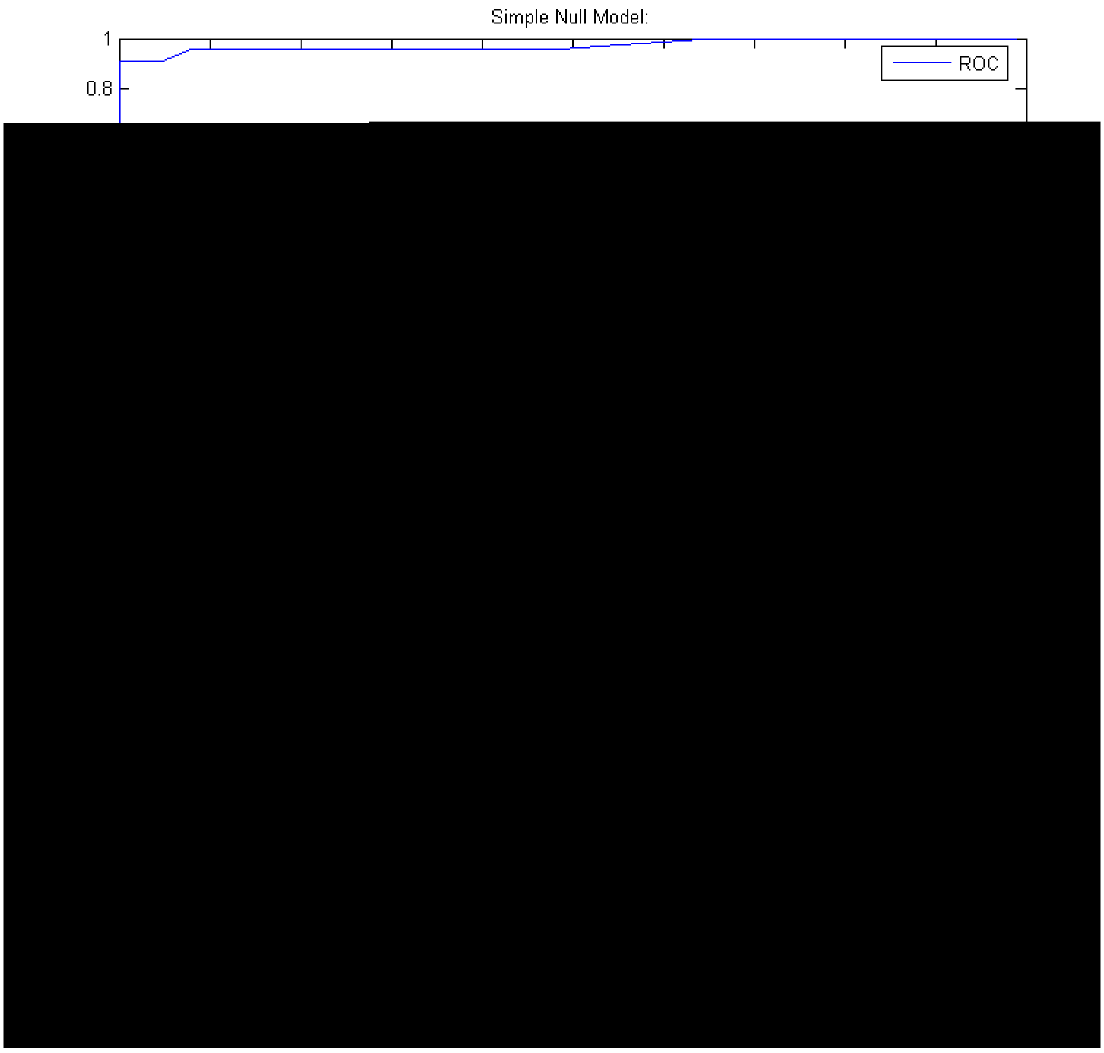


Figure 3.3: Density functions for family a.22.1.3

a value that is classified as positive is in fact a negative one. To classify a value, a threshold is necessary. The FRR and the FAR are calculated for different threshold values and plotted over these thresholds. This method is also used to find a suitable threshold value for the particular test, yet, here they are only used to illustrate the quality of distinctiveness of the different null models. A perfect result would be if the two curves do not intersect at all above a zero, meaning that there are thresholds which imply both an FAR and an FRR of zero.

Quite similar to that is the ROC. It is the function of the Sensitivity, which is equal to 1-FRR, over the FAR. The information given from the ROC is the same as from the FRR-FAR graph, yet the ROC is independent of the threshold values and hence



[redacted]

Figure 3.4: ROC for family a.4.1.1, family distinction

easily comparable. Thus, only the ROC graphs are shown here. The ROC for a perfect disctinction follows the y-axis for an x of zero and constitutes the straight line y = 1 for x greater than zero.

For family distinction, almost every ROC graph has perfect characteristics and those that do not entirely follow that course are very close to it. Regarding model performace, the reverse sequence null model always is closer to the perfect characteristics than the other null models as can be seen in figure 3.4. This reflects the results from the density graphs.

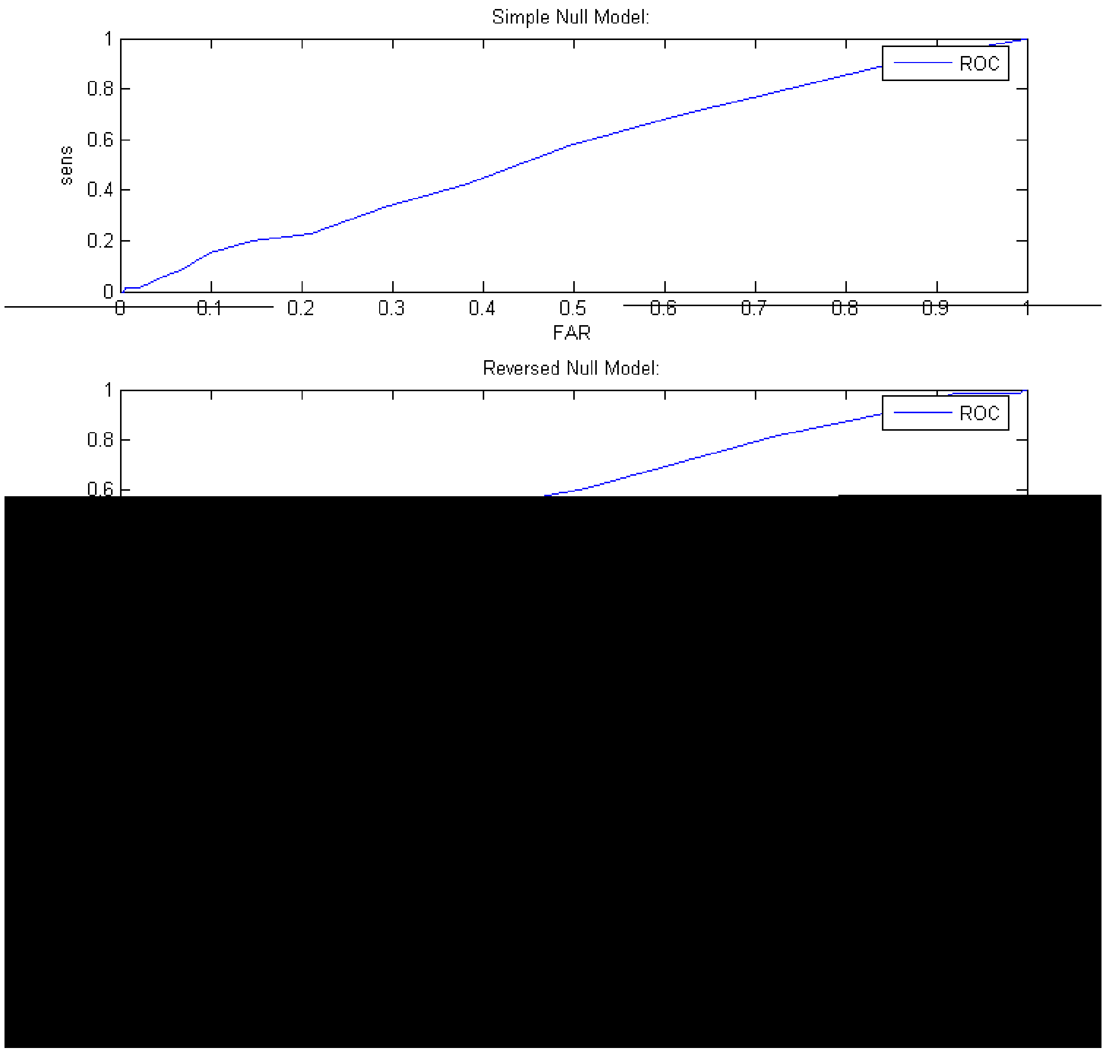
Likewise, for both the super-family and the fold distinction, the ROC reveals a rather poor performance shown in figure 3.5 and 3.6, just as suspected from the density graphs. Furthermore, models' performances vary, showing that no model is better on

Figure 3.5: ROC for family a.4.1.1, fold distinction

a constant basis. However, the ROC too shows a high correlation between the simple and the null2 null model.

3.2.4 Speed

As already assumed during the introduction of the null models, the reverse sequence null model is a lot slower than the other too models, which are equally fast. The calculation time for the reverse sequence null model is twice the calculation time for the other models. This discrepancy has to be considered on further updates of the implementation. A threshold score above which the reverse sequence null model will be triggered, could save time without deteriorating the results.



[redacted]

Figure 3.6: ROC for family a.4.5.28, super-family distinction

3.2.5 Outcome

Considering all data for the tested database and families, the results are clear. The distinction of family members works very good with any null model, with slightly better results from the reverse sequence null model. Considering super-family and fold distinction, no model performs well at all. The high similiarity between the simple null model and the null2 null model can be deducted from the null2 algorithm, which only alters the score for certain cases. To really see the effect of the null2 model, single sequences and their scores have to be looked at, as the null2 model only corrects the score on rare occasions. The better performance of the reverse sequence null model at family distinction leads to the assumption that it corrects inflated scores and false positives caused by biased composition. However, to fully prove this, further investigation is needed, too. Without this further examination, no statement about which model is to prefer can be taken, especially if the speed aspect is taken into account.

4 Conclusion and Outlook

Concludingly, the search for and the implementation of null models more sophisticated as the standard simple null model was a success. Concerning null models, the software is now at least at the same level as the two competitive produchts SAM and HMMer. To grasp the full extent of the improvements, further and more thorough examination with other databases and more family parameter sets is needed. Especially the impact of expert knowledge inserted into parameter sets, which is one of the particular feature of this software suite, in correlation with the new null models has to be investigated.

Following this paper and the preliminary work, there will be a diploma thesis that picks up the subject. The code and the full scale of results of this work will be published with it. However, the further examination talked about will not be part of that piece of work. It will rather deal with the question how the coherrence of the three-dimensional structure of proteins can be transformed into parameters for the profile-HMM. However, if there are any new insights concerning the performance of the null models, it will be mentioned within that piece of work.

Bibliography

[1] Barrett, C., Hughey R. and K. Karplus: *Scoring hidden markov models.* CABIOS (now BIOINFORMATICS), **13**(2):191–199, April 1997.

[2] Durbin, R. et al.: *Biological Sequence Analysis.* Cambridge University Press, 1998.

[3] Eddy, S.: *HMMer User Guide.* An online version is available at `http://www.csb.yale.edu/userguides/seq/hmmer/docs/index.html` (03.02.2009), December 1998.

[4] Hughey, R., Karplus K. and A. Krogh: *SAM Documentation.* An online version is available at `http://compbio.soe.ucsc.edu/papers/sam_doc/sam_doc.html` (03.02.2009), July 2003.

[5] Karplus, K. et al.: *Calibrating e-values for hidden markov models using reverse-sequence null models.* BIOINFORMATICS, **21**(22):4107–4115, November 2005.

[6] Radlingmaier, M.: *Optimierte Hidden Markov Modelle für das Sequenz-Scoring auf Basis von Multiplen Alignments.* Diplomarbeit, Fachhochschule Salzburg, 2007.

List of Abbreviations

ROC	Receiver Operator Characteristic
FAR	False Acceptance Rate
FRR	False Rejection Rate
HMM	Hidden Markov Model
MSA	Multiple Sequence Alignment
SAM	Sequence Alignment and Modeling System