UNIVERSITY OF Southampton
School of Biological Sciences

UniFake: Generation of fake UniProt DAT files

Richard J. Edwards © 2008.

Contents

Tables

1. Introduction

This manual gives an overview of UniFake, a utility for generating a "fake" UniProt flat-file format DAT file from a protein sequence file. UniFake can utilise a number of *in silico* prediction methods to add annotation and additional information can be added through use of a features table. This table is compatible with the features table output of RJE_UNIPROT, allowing real and computer-generated annotation to be combined in one file.

UniFake is designed as a standalone application but additional python modules will need to be present for it to work. For full functionality, additional utilities must be installed; some of these require licenses and/or permissions of the authors. This document should describe these features but if anything is missing or needs clarification, please contact me. The fundamentals are covered in Chapter 2, Fundamentals, including input and output details. Later sections give more details on how the methods work and statistics are generated. General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to UniFake can be found in the distributed readme.txt and readme.html files

Like the software itself, this manual is a 'work in progress' to some degree. If the version you are now reading does not make sense, then it may be worth checking the website to see if a more recent version is available, as indicated by the Version section of the manual. Furthermore, many options have been added to UniFake over the past few weeks and not all of them have found their way into the manual yet. Check the readme on the website for up-to-date options etc. In particular, default values for options are subject to change and should be checked in the readme.

Rich Edwards, 2008.

1.1. Version

This manual is designed to accompany UniFake version 1.2.

The manual was last edited on 10 October 2008.

1.2. Using this Manual

As much as possible, I shall try to make a clear distinction between explanatory text (this) and text to be typed at the command-prompt etc. Command prompt text will be `written in Courier New` to make the distinction clearer. Program options, also called 'command-line parameters', will be **`written in bold Courier New`** (and coloured **`red`** for fixed portions or **`dark`** for user-defined portions, such as file names etc.). Command-line examples will be given in *`italicised Courier New`*. Optional parameters will (if I remember) be `[in square brackets]`. Names of files will be marked in coloured normal text.

1.3. Why use UniFake?

UniFake has two main uses:

1. Simple conversions of plain protein sequence files (e.g. fasta format) into a UniProt DAT file format, for compatibility with tools that require UniProt format.
2. Annotation of sequence features using *in silico* prediction tools.

1.4. Getting Help

Much of the information here is also contained in the documentation of the Python modules themselves. A full list of command-line parameters can be printed to screen using the **`help`** option, with short descriptions for each one:

```
python unifake.py help
```

General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to UniFake can be found in the distributed readme.txt and readme.html files.

If still stuck, then please e-mail me (r.edwards@southampton.ac.uk) whatever question you have. If it is the results of an error message, then please send me that and/or the log file (see Chapter 2) too.

1.4.1. Something Missing?

As much as possible, the important parts of the software are described in detail in this manual. If something is not covered, it is generally not very important and/or still under development, and can therefore be safely ignored. If, however, curiosity gets the better of you, and/or you think that something important is missing (or badly explained), please contact me.

1.5. Citing UniFake

Until published, please cite the UniFake Website. When using predictions from accessory software, please cite the relevant papers.

1.6. Availability and Local Installation

UniFake is distributed as a number of open source Python modules as part of the PEAT (Protein Evolution Analysis Toolkit) package. It should therefore work on any system with Python installed without any extra setup required. If you do not have Python, you can download it free from www.python.org at http://www.python.org/download/. The modules are written in Python 2.5. The Python website has good information about how to download and install Python but if you have any problems, please get in touch and I will help if I can.

All the required files should have been provided in the download zip file. Details can be found at http://bioinformatics.ucd.ie/shields/software/peat/ and the accompanying PEAT Appendices document. The Python Modules are open source and may be changed if desired, although please give me credit for any useful bits you pillage. I cannot accept any responsibility if you make changes and the program stops working, however!

Note that the organisation of the modules and the complexity of some of the classes is due to the fact that most of them are designed to be used in a number of different tools. As a result, not all the options listed in the __doc__() (`help`) will be of relevance. If you want some help understanding the way the modules and classes are set up so you can edit them, just contact me.

2. Fundamentals

2.1. Running UniFake

2.1.1. The Basics

If you have python installed on your system, you should be able to run UniFake directly from the command line in the form:

```
python unifake.py seqin=FILENAME
```

To run with default settings, no other commands are needed. Otherwise, see the relevant sections of this manual.

IMPORTANT: If filenames contain spaces, they should be enclosed in double quotes: *data="example file"*. That said, it is recommended that files do not contain spaces as function cannot be guaranteed if they do.

2.1.2. Options

Command-line options are suggested in the following sections. General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to UniFake can be found in the distributed readme.txt and readme.html files. These may be given after the run command, as above, or loaded from one or more *.ini files (see PEAT Appendices for details).

2.1.3. Running in Windows

If running in Windows, you can just double-click the unifake.py file. Commands should be placed in a file called unifake.ini or rje.ini, which will be read upon execution. It is recommended to use the **win32=T** option.

2.2. Input

UniFake takes a single protein sequence file as input. Acceptable formats include fasta, clustalw alignments, UniProt DAT files, phylip format and fastacmd extraction from a BLAST database. See the RJE_SEQ Manual for details. Any of the sequence filtering/manipulation options available in RJE_SEQ should work on this input file before the main UniFake method is called.

In addition to the main input, UniFake can take additional input files that specify features to add or accession number aliases.

2.2.1. Features Table Input

UniFake can accept one or more simple input tables containing features to be added to each sequence in the file (**features=LIST**). This table should have five columns:

- Acc_Num: The first column should be a sequence identifier matching that of the input sequence file. This can either be a recognisable accession number (see RJE_SEQ) or the first word of the sequence name. (Additional identifiers that do not match the input file will be ignored.)
- Feature: This column should have the "type" of feature, e.g. DOMAIN, MUTATION etc.
- Start: The start position of the feature. (The header ft_start is also accepted.)
- End: The end position of the feature. (The header ft_end is also accepted.)
- Description: The description of the feature.

Multiple files can be given (wildcards are allowed, e.g. **features=*.ft.tdt** will read all files ending .ft.tdt) allowing additional features to be generated by the user from multiple sources and compile them all with UniFake.

To include features from a real UniProt file, first run RJE_UNIPROT on the file using the **ftout=FILE** option, then use the **features=LIST** option of UniFake to read them back in.

Example. UniProt entries have been downloaded in a file test.dat. A new dat file, fake.dat, is to be created containing existing features plus predictions. This needs two commands:

```
python rje_uniprot.py uniprot=test.dat ftout=real_ft.tdt
python unifake.py seqin=test.dat features=real_ft.tdt datout=fake.dat
```

Note that all the additional information in the original file (database links etc.) will be lost in the new file.

2.2.2. Aliases Input

The aliases table (**aliases=FILE**) has a simple format of two columns:

- The first column should contain the sequence identifier matching that of the input sequence file. This can either be a recognisable accession number (see RJE_SEQ) or the first word of the sequence name. (Additional identifiers that do not match the input file will be ignored.)
- A second column headed "Aliases" should contain a comma-separated list of alias accession numbers. Each protein can have several entries in this file and aliases from all relevant file lines will be incorporated.

Recognised aliases will be added to the "AC" line of the DAT file and therefore can be used for extracting sequences with RJE_UNIPROT if an index file is created (**makeindex=T**).

2.3. Output

The main output of UniFake is a DAT file containing the reformatted sequences, with *in silico* predictions as features, if appropriate. Features are listed in order of position. The filename is given by **datout=FILE**. If no filename is given, the input name will be modified to a *.dat name and be used. If the **makeindex=T** option is selected, a uniprot.index file will also be created, indexing any *.dat files in the same directory as the output file. This allows extraction of specific entries using RJE_UNIPROT with the **unipath=PATH** and **extract=LIST** options. (See RJE_UNIPROT for details. Note that this will only work if the output file is named *.dat and the index will include any other *.dat files in the directory.)

2.4. Commandline Options

Table 2.1. UniFake Commandline Options.

Option	Description	Default
	Input Options	
seqin=X	Input sequence file. See rje_seq documentation for filtering options.	[None]
spcode=X	Species code to use if it cannot be established from sequence name.	[None]
feature=LIST	List of files of addtional features in delimited form.	[]
aliases=FILE	File of aliases to be added to Accession number list (for indexing).	[None]
pfam=FILE	PFam HMM download [None]	[None]
	Processing/Output Options	
unifake=LIST	List of predictions to add to entries tmhmm/signalp/disorder/pfam.	[tmhmm,signalp ,disorder,pfam]
datout=FILE	Name of output DAT file.	[*.dat]
disdom=X	Disorder threshold below which to annotate PFam domain as "DOMAIN"	[0.0]
makeindex=T/F	Whether to make a uniprot index file following run.	[False]
ensdat=T/F	Look for acc/pep/gene in sequence name.	[False]
tmhmm=FILE	Path to TMHMM program.	[None]
signalp=FILE	Path to SignalP program.	[None]

Commandline options are given in the appropriate sections. A full list of commandline options (Table 2.1) can be found in the readme file or by running: `python unifake.py help`

The commandline options for rje_seq, rje_disorder, rje_hmm, rje_tm and rje_uniprot might also be of interest.

2.5. Accessory Applications

It is simple to add features generated from accessory applications using the feature=LIST input of tables (see 2.2.1). In addition, UniFake has a number of inbuilt capabilities for using *in silico* predictions. These should be specified in the unifake=LIST option and must be installed on the system. Only limited descriptions are given here – please contact me if you need more details or help getting them working with UniFake.

2.5.1. Disorder

Disorder is controlled by the RJE_DISORDER module. By default, UniFake with use IUPred (Dosztanyi *et al.* 2005) (installation specified by **unipath=PATH**) with the "short" method and a cut-off of 0.2. These options may be altered (Table 2.2). If IUPred is correctly installed and the IUPred_PATH environment variable is set, no other options are needed. If not (and/or this is meaningless gibberish to you) try using the **iuchdir=T** option.

By default, each residue is treated individually and "regions" of order or disorder may be as small as 1aa long. To alter this, use the **minregion=X** option, to specify the minimum length of a region. Small regions are removed in size order, working N-terminal to C-terminal for each length. (e.g. if `minregion=3` then all 1aa regions are removed, then all 2aa regions are removed, starting with the N-terminus in each case.)

Table 2.2. RJE_DISORDER Commandline Options.

Option	Description	Default
disorder=X	Disorder method to use (iupred/foldindex/parse).	[iupred]
iucut=X	Cut-off for IUPred results.	[0.2]
iumethod=X	IUPred method to use (long/short).	[short]
iupath=PATH	The full path to the IUPred executable.	[None]
iuchdir=T/F	Whether to change to IUPred directory and run (True) or rely on IUPred_PATH env variable.	[False]
minregion=X	Min. length for an ordered/disordered region.	[0]

2.5.2. TMHMM transmembrane prediction

TMHMM (Sonnhammer *et al.* 1998) can be used for transmembrane prediction. Specify the path to this program using **tmhmm=FILE**.

2.5.3. SignalP signal peptide prediction

SignalP (Nielsen *et al.* 1997) can be used for signal peptide prediction. Specify the path to the program using **signalp=FILE**.

2.5.4. HMMer prediction of PFam domains

Sequences can be searched for PFam domains (Finn *et al.* 2003) using the HMMer search tool (http://hmmer.janelia.org/). Make sure the PFam_ls file is downloaded from PFam and identified with **pfam=FILE**. HMMer must be installed and the path containing the executables (e.g. /home/richard/Bioware/hmmer-2.3.2/src/) specified with **hmmerpath=PATH**.

2.6. Adding Real UniProt Data

From version 1.2, UniFake is also able to add real UniProt data by including "uniprot" in the **unifake=LIST** parameter. To do this, UniFake must be given the path to the UniProt files using **unipath=PATH**, and these files must have been indexed using rje_uniprot (below). UniFake will then try to extract UniProt entries using the ID and Accession number of the input sequence. If an entry is

found, each Feature will be compared for 100% sequence identity and those with exact matches will be added. Where there is not a match at the position specified, UniFake will try to "fudge" the match by checking increasing distances away from the annotated position and will then annotate the closest found match. This fudging can be switched off using **fudgeft=F**. If no sequence match is found, the feature will not be added.

In addition to Features, UniFake will also add all the DAT file data types specified by **unireal=LIST**. By default, this is: 'AC','GN','RC','RX','CC','DR','PE' and 'KW'.

2.6.1. Indexing UniProt files with rje_uniprot

This module is available as part of the UniFake download. The following command should index UniProt DAT files for use with UniFake:

```
python rje_uniprot.py makeindex=T unipath=PATH
```

You must have rje_uniprot version 3.3 or later. (Please contact the author for details if this is not clear how to do this or this does not work.)

3. Appendices

3.1. Troubleshooting & FAQ

There are currently no specific Troubleshooting issues arising with UniFake. Please see general items in the PEAT Appendices document and contact me if you experience any problems not covered.

3.2. References

Dosztanyi Z, Csizmok V, Tompa P & Simon I (2005). "Iupred: Web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content." *Bioinformatics*. **21**(16): 3433-4.

Finn R, Griffiths-Jones S & Bateman A (2003). "Identifying protein domains with the pfam database." *Curr Protoc Bioinformatics*. **Chapter**(2): Unit 2 5.

Nielsen H, Engelbrecht J, Brunak S & von Heijne G (1997). "A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites." *Int J Neural Syst*. **8**(5-6): 581-99.

Sonnhammer EL, von Heijne G & Krogh A (1998). "A hidden markov model for predicting transmembrane helices in protein sequences." *Proc Int Conf Intell Syst Mol Biol* **6**: 175-82.