## NAME

ffpry − This program generates an FFP vector of nucleic acid RY-coded features.

## SYNOPSIS

**ffpry [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options the default behavior of the program is to generate a Feature Frequency Profile (FFP) using features of length 10 from FASTA input. The FFP will contain the counts of nucleic acid features coded in purine(R)/pyrmidine(Y) format. By default the feature keys will be printed alongside the feature counts. FASTA sequences will be read from standard input if no options are supplied and **ffpry** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list. In default mode features are stored in either the forward or reverse complement direction based upon alphabetical preference.

## OPTIONS

**−l** *LEN*, **--length=***LEN*

Changes the default length of features to *LEN*. The default length is 10. Maximum length allowed in 40. You may overide this maximum length by setting (and exporting) the environmental variable MAX_WORD_SIZE to a different value.

**−f** *FILE*, **--feature-list=***FILE*

Changes the behavior of the program to read a list of features from *FILE*. Features can be space or newline delimited. As of v3.00 features are now matched,coded and reported in both the forward and reverse strand. To disable this behaviour use the **-r** option. By default the feature list is RY coded.

**−w** *STR*, **--mask=***STR*

Use character *STR* of length *LEN* which specifies a character mask for the features. For example "1111101111" means to ignore the 6th character
of a character feature. Feature maskings functions in combination with both **-d** and **-r**.

**−z** *K*, **--rand-mask=***K*

Create a random weight mask which allows up to K mismatches. No greater than the word length in mismatches is allowed. The weight mask is printed to standard error at the beginning of execution. This option is cummulative with both options -d and -r.

**−s** *INT*, **--rand-seed=***INT*

Seed the random number generator, to make randomization operations like **−z** predictable across different runs. This option is cummulative with both options **-d** and **-r**. The defaul seeed to the random number generator is (system time) * (process ID).

**−q, --quiet**

Run in quiet mode. Suppresses printing to standard error.

**−d, --disable**

Disable RY coding to use ATGC coding.

**-m, --multiple**

Calculate FFPs for multiple sequences in the input file. Sequences must have their own FASTA '>' header.

**-r, --disable-rev**

Disable counting of reverse complement features.

## EXAMPLES

The FFP phylogeny tools are designed to be used as filters in a pipeline. To create a key-form FFP use this syntax:

```
ffpry -l 3 test1.fna
```

This will generate an FFP of the form

```
RRY 4 RRR 5 YYR ...
```

To generate a profile of several nucleic acid sequences

```
ffpry -l 3 test1.fna test2.fna test3.fna
```

Or more succinctly

```
ffpry -l 3 test*.fna
```

The profile for each fna file will be produced on a newline delimited row. in the following format:

```
AAATGA 2      ATAGTA 4      ATGGGG 1 ...
AAACGA 2      ATAGCA 1      CTGAGG 3 ...
```

This format is a key-value FFP.

The utility **ffprwn** which row normalizes data must be presented with columnar data -- This data format contains only the count values and no keys. The columns represent a feature and each row of the FFP input corresponds to the counts of that feature in the individual FASTA files. This data format is recognized by several utilities, therefore you must convert a key valued FFP into a columnar format using **ffpcol**

For example several commands can be piped together to produce a divergence matrix representing FFP similarity

```
ffpry -l 4 test*.fna | ffpcol | ffprwn | ffpjsd
```

If mismatches are desired at specific positions within a feature a mask can be applied by using the **−w** option. The format of the mask is a string of 1's and 0's which specify whether a match is required at that postion or whether mismatches are allowed. For example:

```
ffpry -l 5 -w 10110 test*.fna
```

This specifies that mismatches are allowed at the 2nd and 5th postions in every feature.

The **−z** option can be used to randomly create a mask with **n** number of mismatches. The mask is printed to stderr by default. The example below redirects the output of standard error to a file.

```
ffpry -l 5 -z 2 test*.fna 2> mask
```

To disable the default RY coding use the **−d** option which will use 4 base ATGC coding.

If the frequencies for a small set of features is desired then the **−f** *FILE* option can be used. *FILE* specifies the name of a file containing a list of features (RY or ATGC coded). The list of features can be tab, space or newline delimited. Note the length of the features must also be specified using the **−l** option as well.

By default all of the sequence data in a particular file is merged into one FFP. Multiple fasta records (multiple fasta sequences in a single file) are parsed as one single sequence (for example, consider the case of multiple contigs in a single genomes). If your fasta files contain multiple sequences separated by '>' style records headers and you wish to produce FFPs for each record then specify the **-m** option.

## L−MER FEATURES

Features are stored in a mixture of the forward and reverse complement direction. For example, if the feature:

```
GTGTAGT
```

is encountered in the forward direction in the sequence it will be stored by default in the hash table and reported in the output in the reverse complement direction, which is:

```
ACTACAC
```

This decision is based upon hash function precedence, whichever direction has the smallest hash index will be used. This form of hashing allows for homology detection, independent of gene strandedness. The behavior can be disabled with the **−r** option, which will force all features to be stored in the forward direction. Also, by default features are stored in RY coded form. This has been shown to improve phylogenetic signal in a number of studies especially if highly varying rate of mutation are suspected among different taxa. Additionally RY coding is much faster, but is a compromise between speed and sensitivity. Likewise this feature can be disabled with the **−d** option.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffprwn(1), ffpmerge(1), ffpjsd(1), ffpcol(1)**

## NAME

ffpaa − This program generates an FFP vector of amino acid features.

## SYNOPSIS

**ffpaa [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options the default behavior of the program is to generate a Feature Frequency Profile (FFP) using features of length 4. By default amino acids are divided and counted using 11 equivalent classes: (ST),(DE),(KQR),(IVLM),(FWY),C,G,A,N,H,P [Li et al. (2003) Peds, 16. 323]. For multicharacter classes the amino acid is reported in the ffp output by the first character in the series. For example, (IVLM) is represented by I.

## OPTIONS

**−l** *LEN*, **--length=***LEN*

Changes the default length of features to *LEN*. The default length is 4, with a maximum length allowed of 40. You may overide this maximum length by setting (and exporting) the environmental variable MAX_WORD_SIZE to a different value.

**−f** *FILE*, **--feature-list=***FILE*

Changes the behavior of the program to read a list of features from *FILE*. Features can be space or newline delimited.

**−w** *STR*, **--mask=***STR*

Use character *STR* of length *LEN* which specifies a character mask for the features. For example "1111101111" means to ignore the 6th character of a feature.

**−z** *INT*, **--rand-mask=***INT*

Create a random character mask which allows up to *INT* mismatches. No greater than *LEN* in mismatches is allowed. The character mask is printed to standard error at the beginning of execution.

**−s** *INT*, **--rand-seed=***INT*

Seed the random number generator. Randomization operations like **-z** are fixed and predictable across different runs. By default the seed is (system time) * (process ID).

**−q, --quiet**

Run in quiet mode. Suppresses printing to standard error. This effects option **−z**.

**−d, --disable-classes**

Disable classing of amino acids. Use 20 symbol amino acid coding.

**−m, --multiple**

*FILE* contains multiple FASTA sequences and an FFP is desired for each sequence.

**−h, --help**

Display help message.

**−v, --version**

Display version.

## EXAMPLES

The command:

ffpaa -l 3 file*.faa

Will produce a output of this form, using features of length 3:

```
AGK 2 ASS 3 AKF 4 ...
AKK 3 AKS 2 AKF 5 ...
```

There will be one FFP line for each `.faa` file supplied in the argument list and each will be printed in the order specified on the command line. Note, the above uses the order in which the shell (i.e. bash, cshell, etc.) expands the file argument wildcard, In the even this is undesired, the format format below can also be used to explicitly control the ordering:

```
ffpaa -l 4 file1.faa file2.faa file3.faa
```

A mask can be applied to feature counting to allow mismatches using the **−w** command. For example if a mismatch is to be allowed at the second and fourth positions of a feature with length *l*=6, *this can be done via:*

```
ffpaa -l 6 -w 101011 file*.fna
```

If mismatches are desired but no specific pattern is necessary then one can use the **-z** option.

```
ffpaa -z 2 file1.faa file2.faa.
```

This will produce a random mask of at least two mismatches (within a feature with a default length of 4). The mask is printed to standard error. To save the mask for later reference, you can direct standard error to a file. For example:

```
ffpaa -z 3 file*.fna > vector 2> mask
```

The redirection `>2` sends standard error to the file `mask.`

To disable classing of amino acids, (i.e. make FFP consider R and K different -- and therefore count each symbol separately) use the **-d** option

The utilities in the FFP package can be used as fully qualified filters and can be used to build pipelines. The pipeline below will create a Phylip infile format distance matrix, using the Jensen Shannon divergence:

```
ffpaa -l 7 file*.fna | ffpcol -a | ffprwn | ffpjsd -p taxa.txt > infile
```

Note, the desired taxa names must be specified in the argument to **−p**, see **ffpjsd(1)**

To disable classing of animo acids you should incorporate the **−d** option into the above pipeline -- both in the call to **ffpaa** and **ffpcol.**

```
ffpaa -l 7 -d file*.fna | ffpcol -a -d | ffprwn ...
```

Bootstrap and jacknife resampling can be done with the ffpboot program, see **ffpboot(1).**

## FURTHER DIRECTIONS

Command line definable character classes.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpry(1), ffprwn(1), ffpjsd(1), ffpcol(1), ffpmerge(1), ffpboot(1), ffpvprof(1), ffpreprof(1), ffptree(1)**

## NAME

ffptxt − This program generates an FFP vector of text data.

## SYNOPSIS

**ffpaa [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options at all the default behavior of the program is to generate a Feature Frequency Profile (FFP) of text data, using 26 alphabet letters, removing all non-alphabetic characters. The default feature length is 6. Text will be read from standard input if no options are supplied and **ffptxt** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list.

## OPTIONS

**−l** *LEN*, **--length=***LEN*

Changes the default length of features to *LEN*. The default length is 6. The maximum word size is 40. You may overide this maximum length by setting (and exporting) the environmental variable MAX_WORD_SIZE to a different value.

**−f** *FILE*, **--feature-list=***FILE*

Changes the behavior of the program to read a list of features from *FILE*. Features can be space or newline delimited.

**−h, --help**

Display help message.

## EXAMPLES

The command:

```
ffptxt -l 3 file*.txt
```

Will produce a output of this form, using features of length 3:

```
hel 2 elo 3 low 4 ...
thi 3 ism 2 sme 5 ...
```

There will be one FFP line for each .txt file

This format can also be used:

```
ffptxt -l 4 file1.txt file2.txt file3.txt
```

The utilities in the FFP package can be used as fully qualified filters and can be used to build pipelines. This pipeline will create a phylip infile format distance matrix, using the Jensen Shannon Divergence:

```
ffptxt -l 7 file*.txt | ffpcol -t | ffprwn | ffpjsd -p > infile
```

Please note the use of the **-t** option in ffpcol to specify text.

Bootstrap and jacknife resampling can be done with the ffpboot program.

## FUTURE DIRECTIONS

None.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpry(1), ffprwn(1), ffpjsd(1), ffpcol(1), ffpmerge(1), ffpboot(1), ffpvprof(1), ffpreprof(1)**

## NAME

ffpcol - This program creates columnar FFPs from key-value output of ffpry.

## SYNOPSIS

**ffpcol [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options, the default behavior of the program is to convert key-value FFPs to columnar format. This utility is used to convert the output of **ffpry**, **ffpaa**, or **ffptxt** to the format required for **ffprwn** if the FFP has been saved in key-value format. FFPs will be read from standard input if no options are supplied and **ffpcol** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the file argument list. The default input type is a key-value nucleotide FFP. When input is directed from a pipe a temporary file is created in the environmentally defined TMP directory or /tmp if this variable is undefined.

## OPTIONS

**−h, --help**
Display help command.

**−a, --amino**
Input is an amino acid FFP. Default is nucleotide.

**−t, --text**
Input is 26 letter text. Default is nucleotide.

**−d, --disable**
Disable classing of either nucleotide or amino acid sequence.

**−V, --verbose**
Be more verbose.

**−v, --version**
Print version information

**−h, --help**
This help message

## EXAMPLES

A columnar format file produced by **ffpcol** is the main interchange format to pass FFPs to a number of other utilities. Sample input from a feature counter such as **ffpry** may resemble something like this:

```
ATG     2       TGC      3      GGA     4 ...
TGC     1       CAC      2      GGA     1 ...
```

From the key-value input format, **ffpcol** will find all the features contained in all the FFPs and align the feature frequency in the same tab delimited column, printing out zeros if a feature is missing in a particular row. The actual sequence of the feature is discarded at this point. The columnar format files produced from input like that above will look like:

```
2       3       0       4
0       1       2       1
```

To convert a (key,value) FFP for use in **ffprwn,** simply arrange the conversion process as a pipeline, whereby the standard output of one utility is sent to the standard input of another:

```
ffpry -l 2 *.fna | ffpcol | ffprwn
```

To disable classing use this format (take special node of the **−d** switch in both of the calls to **ffpry** and **ffpcol**):

```
ffpry -l 4 -d *.fna | ffpcol -d | ffprwn
```

For amino acids, use the **−a** option:

```
ffpaa -l 4 *.faa | ffpcol -a | ffprwn
```

To disable classing for amino acids use the options **−a** and **−d**, or in stacked form, **−ad**:

```
ffpaa -l 4 -d *.faa | ffpcol -ad | ffprwn
```

Likewise for text input use **−t**:

```
ffptxt -l 6 *.txt | ffpcol -t | ffprwn
```

Note when piping output from a utility into **ffpcol** via a pipe that a temp file is created ( from the output of preceding command ), whereas input from file arguments or via input redirection (i.e. `ffpcol < file` ) doesn't require temp files. Temporary files are written to `/tmp` -- upon creation they are immediately unlinked so they are effectively 'hidden' temp files.

## FUTURE DIRECTIONS

Conversion to Rabin-Karp Hash function.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpry(1), ffpaa(1) ffptxt(1) ffprwn(1), ffpjsd(1)**

## NAME

ffprwn − This program performs row normalization of an FFP vector file.

## SYNOPSIS

**ffprwn [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options, the default behavior of the program is to generate row normalized relative frequency vectors. Input should be a columnar (not a key,value) FFP. Use **ffpcol** to convert to columnar format beforehand. FFPs will be read from standard input if no options are supplied and **ffprwn** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list.

## OPTIONS

**−n, --largest-row**

Alternate form of row normalization that normalizes by the FFP row with the largest sum. Note this will only normalize within individual files specified on the command line, not all FFPs in all files. If all elements of a row are zero, then each element will have a relative frequency of zero after normalization.

**−d** *INT***, --precision=***INT*

Specify *INT* digits of decimal precision. The default is 2

**−h, --help**

Display help message.

## EXAMPLES

Here is an example of how to use **ffprwn** in a pipeline with **ffpry**. To handle key-value paired FFP input use the **ffpcol** utility as a filter.

```
ffpry -l 5 test*.fna | ffpcol | ffprwn | ffpjsd > matrix
```

Likewise amino acid sequences can be handled in a similar fashion.

```
ffpaa test*.fna | ffpcol -a | ffprwn | ffpjsd > matrix
```

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1) ffpry(1), ffpboot(1), ffpjsd(1), ffpcol(1), ffpmerge(1), ffpvprof(1), ffpreprof(1)**

## NAME

ffpboot − This program performs bootstrap permutations of the FFP vector.

## SYNOPSIS

**ffpboot [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options, the default behavior of the program is to calculate a bootstrap permutation. This utility should be used as a filter before using **ffprwn.** FFPs will be read from standard input if no options are supplied and **ffpboot** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list.

## OPTIONS

**−j, --jackknife**

Perform deletion jacknife instead of bootstrap. Default deletion percentage is 1/e.

**−p** *FLOAT***, --delete-prob=***FLOAT*

Specify jacknife deletion Probabilitiy, *FLOAT*, ranging between 0 and 1.

**−s** *INT***, --rand-seed=***INT*

Specify random seed, *INT*. The default is (system time) * (process ID).

**−h, --help**

Display help message.

## EXAMPLES

To create bootstrap pseudoreplicate:

```
ffpry -l 6 test*.fna | ffpcol > vector
ffpboot vector > boot
```

To create many pseudoreplicates use the shell's looping facilities. This example below creates 100 psuedoreplicates, using a small snippet of Bash shell scripting -- the equivalent should be possible using all other shell variants (c-shell, zsh, etc.)

```
for i in $(seq 1 1 100) ; do
        ffpboot vector > boot.$i
done
```

The bootstrap pseudoreplicates can be used as input to **ffprwn**:

```
ffpboot vector | ffprwn | ffpjsd
```

To perform a deletion jackknife instead use the **-j** option.

```
ffpboot -j -p 0.10 vector > jack
```

The above example sets the probability of feature deletion in the jackknifing process to 10%.

As of version 3.06, there is a streamlined method of generating pseudo-replicate trees, which can be used in the phylip program `consense` to build a consensus tree.

```
for i in {1..100} ; do
        ffpboot vector | ffprwn | ffpjsd -p species.txt | ffptree -q
done > intree
```

The final output `intree` contains multiple sets of Newick format tree files which can be used directly in `consense`.

## FURTHER DIRECTIONS

Option to create multiple replicates in a single exection.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1), ffprwn(1), ffpjsd(1), ffpcol(1), ffpmerge(1), ffpvprof(1), ffptree(1), ffpreprof(1)**

## NAME

ffpjsd − Calculates a Jensen Shannon divergence matrix from a row normalized vector FFP.

## SYNOPSIS

**ffpjsd [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options, the default behavior of the program is to generate a symmetric Jensen Shannon divergence matrix. Rather than this divergence, other metrics can be used with the appropriate options. FFPs will be read from standard input if no options are supplied and **ffpjsd** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list. Row normalization with **ffprwn** is required to use the default metric, the Jensen Shannon divergence. Other distance metrics such as the continuous distance measures can be used with or without row normalization with different effects. Row normalization is not necessary with binary distances and has no effect.

## OPTIONS

**−p** *FILE*, **--phylip=***FILE*

Creates a phylip format 'infile'. FILE specifies the taxon names to use. Must be equal to the number of FFP vectors rows. Note the taxon names should be unique. Names over 10 characters in length will be truncated to exactly 10 -- truncation is enforced to maintain compatiblility witht Phylip package.

**−d** *INT*, **--precision=***INT*

Specify INT digits of decimal precision, the default is 2.

**−r** *ROW*, **--row=***ROW*

Specify a single row of the distance matrix to calculate. This option is useful for especially large matrices where the different rows can be calculated in a multi-processor environment. Currently only works with the Jensen Shannon divergence metric.

**-s, --similarity**

Print a similarity matrix rather than a distance matrix. This option effects the output of distances metrics which have a value normalized from 0 to 1 or -1 to 1, which includes the metrics specified by the options: **−RcmjtdNSaPBguyok.**

**-h, --help**

Display help message.

**-v, --version**

Display version information.

### CONTINUOUS DISTANCE MEASURES

**-e, --euclid**

Calculate a Euclidean distance matrix rather than the default JSD matrix. Node, that the norm of the distance can by changed with the **−n** option, however the default norm is 2.

**-E, --euclid2**

Calculate a Euclidean squared distance matrix.

**−n** *FLOAT*, **--normval=***FLOAT*

For option **-e**, change the n-norm distance (Default is n=2) to any other value where n > 1

**-c, --cosine**

Calculate a Cosine distance matrix rather than default JSD matrix. With option **-s** this is the similarity matrix.

**-m, --manhattan**

Calculate a Manhattan distance matrix rather than default JSD matrix.

**-R, --pearson**

Use the pearson correlation coefficient. With the **-s** option a similarity matrix will be printed out. Note this is R not, R_squared. Otherwise a distance will be printed, which is 1-R_squared.

**-C, --chebyshev**
Compute the Chebyshev distance, which is the maximum difference between a pair of features.

**-b, --canberra**
Compute the Canberra distance matrix.

**-H, --hamming**
Compute the hamming distance.

**-L, --evol**
Compute the Evolutionary Distance used in E.coli Publications.

**BINARY DISTANCE MEASURES**

With these options the input FFPs are treated as binary data. When two FFPs (i and j) are compared each distance measure uses a cross tabulation for pairwise feature comparison with sums A, B, C and D. A is the number of features which are present in both vectors while D is the number of features that are absent in both vectors. B means the feature is present in i and absent in j. C means the feature is absent in i but present in j. N is the sum of A+B+C+D. All of the binary distance options can be used together with the **-s** option to print a similarity matrix. THe binary distance do not need to be normalized with **ffprwn.**

**-M, --matching**
Compute the matching distance matrix, which is 1-(A+D)/N. With the additional option **-s** this is the matching similarity: (A+D)/N.

**-j, --jaccard**
Compute the Jaccard distance matrix, which is 1-A/(N-D). With the additional option **-s** this is the Jaccard similarity: A/(N-D)

**-t, --tanimoto**
Compute the Rogers-Tanimoto distance matrix, which is 1-(A+D)/((A+D)+2(B+C)). With option **-s** this is the Tanimoto similarity matrix: (A+D)/((A+D)+2(B+C)).

**-D, --dice**
Compute the Dice distance matrix, which is 1-2A/(2A+B+C). With option **-s** this is the Dice similarity: 2A/(2A+B+C).

**-N, --antidice**
Compute the anti-Dice distance matrix, which is 1-A/(A+2(B+C)). With option **-s** this is the anti-Dice similarity: A/(A+2(B+C)).

**-S, --sneath**
Compute the Sneath-Sokal distance matrix, which is 1-2(A+D)/(2(A+D)+(B+C)). With option **-s** this is the similarity matrix: 2(A+D)/(2(A+D)+(B+C)).

**-a, --hamman**
Compute the Hamman distance matrix, which is 1-[((A+D)-(B+C))/N]^2. With option **-s** this is the similarity matrix: ((A+D)-(B+C))/N, which ranges from -1 to 1.

**-P, --phi**
Compute the Pearson Phi distance matrix, which is 1-[(AD-BC)/sqrt((A+B)(A+C)(D+B)(D+C))]^2. With option **-s** this is the similarity matrix: (AD-BC)/sqrt((A+B)(A+C)(D+B)(D+C)), which ranges from -1 to 1.

**-B, --anderberg**
Compute the Anderberg distance matrix, which is 1-(A/(A+B)+A/(A+C)+D/(C+D)+D/(B+D))/4. With option **-s** this is the similarity matrix: (A/(A+B)+A/(A+C)+D/(C+D)+D/(B+D))/4.

**-g, --gower**
Compute the Gower distance matrix, which is 1-AD/sqrt((A+B)(A+C)(D+B(D+C)). With option **-s** this is the similarity matrix: AD/sqrt((A+B)(A+C)(D+B(D+C)).

**-u, --russel**
Compute the Russel-Rao distance matrix, which is 1-A/N. With option **-s** this is the similarity matrix: A/N.

**-y, --yule**
Compute the Yule distance matrix, which is 1-[(AD-BC)/(AD+BC)]^2. With option **-s** this is the similarity matrix: (AD-BC)/(AD+BC) which ranges from -1 to 1.

**-o, --ochiai**
Compute the Ochiai distance matrix, which is 1-A/sqrt((A+B)(A+C)). With option **-s** this is the similarity matrix: A/sqrt((A+B)(A+C)).

**-k, --kulczynski**
Compute the Kulczynski distance matrix, which is 1-(A/(A+B)+A/(A+C))/2. With option **-s** this is the similarity matrix: (A/(A+B)+A/(A+C))/2.

## EXAMPLES

This utility is used as the final filter and can produce a final distance matrix representing FFP based sequence similarity.

```
ffpry -l 5 test*.fna | ffpcol | ffprwn | ffpjsd > matrix
```

Using the **-p** option, **ffpjsd** can be used to create Phylip format infile output.

```
ffpaa -l 4 test*.faa | ffpcol -a | ffprwn | ffpjsd -p species.txt > infile
```

The Phylip package's distance based tree methods can be used to produce phylogenetic trees. Note that `species.txt` should contain the names of the taxa that correspond to the rows of the matrix. As of version 3.06, **ffptree** is included in the FFP package which will create neighbor joining or UPGMA trees. It can also be included in the general ffp pipeline to produce final tree output in Newick format. The pipeline below produces a neighbor joining tree:

```
ffpaa -l 4 test*.faa | ffpcol -a | ffprwn |
ffpjsd -p species.txt | ffptree -q > tree
```

## FURTHER DIRECTIONS

Extend row based -r option to distance measures other than JSD.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1), ffprwn(1), ffpboot(1), ffpcol(1), ffpmerge(1), ffpvprof(1), ffptree(1), ffpreprof(1)**

## NAME

ffpdf − Outputs clade distinguishing features for specified clades.

## SYNOPSIS

**ffpdf [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

This is used to discover distinguishing (or diagnostic) features (DFs) that are present in a specific clade and no other clades. This technique was used in the paper: Sims GE, and Kim SH (2010), PNAS, 108, 8329-34. This ultility should be used as a filter for the output of **ffpry**, **ffpaa**, or **ffptxt** using key-valued FFPs input. Given no file arguments, the FFP input can be piped into STDIN. If a group (i.e. clade) file is specified but no group is specified then, the group file is ignored. You must query individual groups separately. Output is a tab delimited list of the features.

## OPTIONS

**−f** *FILE*, **--group-file=***FILE*

Specify clades/groups. FILE is a newline delimited file which indicates which row of the FFP file belongs to which clade. FILE has a format specifying one newline delimited symbol per row. See **EXAMPLES** below. If not specified then all rows are assumed to be in the same clade. Use in conjunction with option **−g**.

**−g** *STR*, **--group=***STR*

Select the clade where the DF search will be performed. If not specifed all rows of the FFP are assumed part of one clade.

**−p** *FLT*, **--percent=***FLT*

Specify a fractional cutoff for the percentage ( i.e. 50% = 0.5) of a group members which which need to have a feature for it to be considered a DF. The default is all group members or 1.0.

**−v, --version**

Display version information.

**−h, --help**

Display help message.

## EXAMPLES

A distinguishing feature is a feature which is present in all FFPs of a given group, but absent in all other groups. Given these example rows from an FFP matrix,

```
Taxon 1 ... ATG 3 CAC 2 TGA 3 ...
Taxon 2 ... ATG 2 CAC 1 TTG 1 ...
Taxon 3 ... ATG 4 TGC 3 GGA 2 ...
```

and the additional information that taxa 1 and 2 belong to group A and taxon 3 belongs to group B, we can give an example of a DF. The feature, `ATG` ,is not a DF for either group A or B, since it is present in all groups. However feature `CAC` is a DF for group A, since it is present in only group A and not group B. Likewise feature `TGA` and `TTG` are not DFs of group A because neither is present in all members of that group.

To find distinguishing features in an FFP, first you must create an FFP, using one of the feature counting utilities, **ffpry**, **ffpaa**, or **ffptxt**. The example below illustrates the method for RY-coded nucleotide sequences, where we want to find features which are universally shared among all taxa specified in the FFP input file.

```
ffpry -l 6 test*.fna > vector
ffpdf vector > dfs
```

or alternatively, combining the separate commands as one pipeline:

```
ffpry -l 6 test*.fna | ffpdf > dfs
```

The above example assumes every row (genome, species, or sequences) is a member of the same group (clade). For multiple clades, use the **--group-file** option to specify the clade structure. This is defined by

building a newline delimited group-file definition. Lets say there are 5 genomes and there are two clades, the 2nd and 3rd rows of the FFP, belong to clade A and the remainder belong to clade B., then the group file should have the format below

```
$ cat groupfile
B
A
A
B
B
```

Now that clades have been defined we can extract out the DFs for clade A or B separately (A in this case below).

```
ffpry -l 6 test*.fna | ffpdf --group-file="groups.txt" --group="A" > dfs.A
```

Multi-character alphanumeric symbols can also be specified for group names. Note however, any white-space will be stripped from group names.

Also, note that by default a DF is defined as a feature which is present in all members of a group. This defintion can be relaxed slightly by using the **−p** option to specify the percentage of taxa within a given group which must have a feature in order for it to be considered a DF. For example if we wanted to find out which features are present in 2 out of 3 taxa in group B we can use this syntax:

```
ffpry -l 6 test*.fna | ffpdf -p 0.66 --group-file="groups.txt" --group="A" > dfs.B
```

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1)**

## NAME

ffpre − This program calculates the relative entropy between an $l-2$ Markov model estimation of frequency and the observed frequency of features. This is used to calculate an upper limit of $l$ to use for phylogenetic inference.

## SYNOPSIS

**ffpre [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options the default behavior of **ffpre** the progam is to generate the relative entropy value (RE) (also known as the Kullbach-Leibler divergence) of feature frequencies using $l$=10 and estimates of the frequencies using $l$=9 and $l$=8 to provide the estimate. To calculate the relative entropies for a range of $l$, use the shell script supplied **ffpreprof** with the FFP package, which calculates a relative entopry profile. This utility is used to help determine the proper feature length range to use for phylogenetic inference. When the relative entropy approaches zero, this indicates that the frequency of all features can be determined by using the frequencies of the $l$-1 and $l$-2 feature frequencies (i.e. features have the same frequency as slightly shorter words). This indicates that each $l$, $l$-1, and $l$-2 feature occurs in the same context within the sequence. Therefore using lengths longer than when the RE is zero provide no additional information. The default assumes nucleic acid sequence input.

## OPTIONS

**−l** *LEN*, **--length=***LEN*

Changes the default length of features to *LEN*. The default length is 10. Maximum length allowed in 40.

**−d, --disable-ry**

Disable RY coding to use ATGC coding.

**−r, --no-reverse**

Disable reverse complement matching. Applicable to nucleotide sequence only.

**−a, --amino**

Input is amino acid sequence.

**−t, --text**

Input is a text file.

**−v, --version**

Print program version.

**−h, --help**

Help message.

## EXAMPLES

To calculate the relative entropy between the estimate and observed feature frequencies:

```
ffpre -l 4 test*.fna
```

To calculate the RE for a range of lengths use **ffpreprof**. Note that the RE may oscillate above or below zero - but given long enough $l$ it will eventually approach zero. In mathematical terms the limit of $KL(l)$ as $l$ approaches infinity is zero.

## FUTURE DIRECTIONS

None.

## AUTHOR

This program was written by Gregory E. Sims

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpvprof(1), ffpvocab(1), ffpreprof(1)**

## NAME

ffpsubnam − Substitute taxa names in Phylip format infile.

## SYNOPSIS

**ffpsubnam [***OPTIONS***] [***NAMES***] [***INFILE***]**

## DESCRIPTION

This is a simple utility to change the names in a phylip format infile for those supplied by an input file.

## OPTIONS

**−o** *FILE*, **--outfile=***FILE*

Specify output file. Default is standard out.

**−v, --version**

Display version information.

**−h, --help**

Display help message.

## EXAMPLES

None.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1)**

## NAME

ffpvocab − This program determines the number of features which have frequencies above a threshold value.

## SYNOPSIS

**ffpvocab [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Given no options, the default behavior of the program is to print out the average number of features with frequencies greater than two in all rows. This value is defined as the number of vocabulary features. FFPs will be read from standard input if no options are supplied and **ffpvocab** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list. If a multi-row FFP is supplied then the default behavior is to average the number of vocabulary features across all rows.

## OPTIONS

**−f** *INT*, **--freq-thresh=***INT*

Threshold to count a feature. The default value is 2, which means a feature must have a frequency greater than or equal to 2, in order to be counted.

**−h, --help**

## EXAMPLES

To count the number of features that occur 3 or more times in the ffp:

```
ffpry -l 5 file.fna | ffpvocab -f 3
```

To calculate word usage for a range of lengths use **ffpvprof.**

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpvprof(1), ffpre(1), ffpreprof(1)**

## NAME

ffpcomplex - This program filters FFP profiles based on the complexity of features.

## SYNOPSIS

**ffpcomplex [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

This utility can be used to filter high or low complexity features from the FFP matrix. The upper and/or lower limits can be set using a complexity cuttoff (the default) or a probability cutoff can be used assuming a normal distribution. Probability limits are the cummulative probability values from the normal distribution found from the calculated mean and stdev of the input FFP. Requires key-valued FFP input. FFPs will be read from standard input if no options are supplied and **ffpcomplex** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list. The default input type is assumed to be nucleotide sequence.

## OPTIONS

**-s, --stats**

Print statistics describing the FFP complexity distribution to standard error.

**-u** *FLOAT,* **--upper=***FLOAT*

The upper threshold limit for feature complexity where *FLOAT* can be any complexity value or for probability distributions, in which case *FLOAT* can range from 0 to 1 -- representing the probability in a cummulative distribution function. This can be used in conjunction with **-l** or by itself.

**-l** *FLOAT,* **--lower=***FLOAT*

The lower threshold limit. This can be used in conjuntion with **-u** or by itself.

**-n, --norm**

Use a normal distribution for filtering. Default is the raw complexity values.

**-d, --disable**

Disable classing of amino acid and nucleotide sequences.

**-a, --amino**

Input FFPs are animo acid sequences.

**-t, --text**

Input FFPs are text.

**-h, --help**

Display short help message.

**-v, --version**

Display version information.

## EXAMPLES

The main usage of this progam is for filtering out low complexity features, for example features which primarily represent repetitive DNA might be need to be filtered out -- since they don't provide much useful phylogenetic signal. To get an idea about what the complexity of the words in your FFPs are you can use the **−s** option to get a statistical report, which includes the mean, range and standard deviation of complexity. The complexity of a feature is determined by finding the total entropy of all individual sub features for all *k* less than *l*, which is the length of the feature. For example given the feature

```
GCGCGCGC
```

determine the entropy of *k*=1, which which is H(1) = f(G)log f(G) + f(C)log f(C). Next determine the entropy of all sub-mers, H(2) = f(GC)log f(GC) + f(CG)log f(CG). Repeat for all *k* less than *l*. The total complexity of the *l*-mer is H(l) = H(1) + H(2) + ... + H(l-1). Here we filter out all features with complexity less than 1.0:

```
ffpry -l 5 *.fna > vectors
ffpcomplex -l 1.0 vectors > vectors.filt
```

Features with high and low complexities can be removed simultaneously. This example excludes all features which with complexity greater than 10 and less than 2.

```
ffpcomplex -u 10.0 -l 2.0
```

Finally we can assume that the complexities of the features used follow a specific distribution (usually the normal distribution is the appropriate choice). We can exclude features that lie outside a certain range of the cummulative probability distribution. Here we exclude all features which lie below 10% and above 95% of the normal CDF.

```
ffpcomplex -n -l 0.1 -u 0.95
```

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpry(1), ffpfilt(1)**

## NAME

ffpfilt - This program filters FFP profiles based on numeric/statistical properties.

## SYNOPSIS

**ffpfilt [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

This utility can be used to filter high or low frequency features from the FFP matrix. The upper and/or lower limits can be set using a raw frequency cutoff or a probability cutoff which assumes either a normal or an extreme value distribution. Requires key-valued FFP input. Note you should use this as a replacement for **ffpcol** in any pipelines (rather than in addition to) as **ffpfilt** produces columnar output. Columnar output can be disabled with the **--keys** option. FFPs will be read from standard input if no options are supplied and **ffpfilt** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list. The default input is assumed to be nucleotide sequence.

## OPTIONS

**-s, --stats**

Print statistics describing the FFP frequency distribution to standard error.

**-z, --zeros**

Include zeros is calculation of statistical parameters. When few features are observed, but a long feature length is used this has the effect of skewing the feature distribution towards zero.

**-u** *FLOAT,* **--upper=***FLOAT*

The upper threshold limit for raw frequencies where *FLOAT* can be any frequency value or for probability distributions, in which case *FLOAT* can range from 0 to 1.

**-l** *FLOAT,* **--lower=***FLOAT*

The lower threshold limit.

**-f, --freq**

Use raw frequencies for filtering

**-n, --norm**

Use a normal distribution for filtering

**-e, --evd**

Use an extreme value (Gumbel) distribution for filtering

**-d, --disable**

Disable classing of amino acid and nucleotide sequences.

**-a, --amino**

Input FFPs are animo acid sequences.

**-t, --text**

Input FFPs are text.

**-k, --keys**

Force key,value output. Default it columnar output.

**-h, --help**

Display short help message.

**-v, --version**

Display version information.

## EXAMPLES

The main usage of this progam is for filtering out high frequency features, for example large numbers of prophage copies in a genome might be need to be filtered out -- since they don't provide much useful phylogenetic signal. Here we filter out all features which occur more than 5 times using the **−u** option.

```
ffpry -l 5 *.fna > vectors
ffpfilt -f -u 5 vectors > vectors.filt
```

Features which occur rarely and features which occur frequently can be removed simultaneously. This example excludes all features which occur more than 10 times and less than 2 times.

```
ffpfilt -f -u 10 -u 2
```

Finally we can assume that the feature frequencies follow a specific distribution. Usually the extreme value distribution [EVD] is the appropriate choice, however you should confirm what the distribution of your feature frequencies are yourself. We can exclude features that lie outside a certain range of the cummulative probability distribution. Here we exclude all features which lie below 10% and above 95% of the EVD cummulative distribution function.

```
ffpfilt -e -l 0.1 -u 0.95
```

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1), ffprwn(1), ffpjsd(1), ffpcol(1)**

## NAME

ffpmerge - This program merges the rows of an FFP into a single row.

## SYNOPSIS

**ffpmerge [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

The behavior of the program is to merge all rows of the FFP. The FFP can be a columnar or key-valued FFP. FFPs will be read from standard input if no options are supplied and **ffpmerge** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list.

## OPTIONS

**-d, --disable**
Disable classing of amino acid and nucleotide sequences.

**-k, --keys**
Print key value pairs. Default is output in columnar format.

**-a, --amino**
Input FFPs are animo acid sequences.

**-t, --text**
Input FFPs are text.

**-v, --version**
Display version.

**-h, --help**
Display help message.

## EXAMPLES

The main usage of this program is to merge separately calculated FFPs from different parts of a large genome. For example consider a multi-chromosome genome.

```
ffpry -l 5 chr1.fna > chr1.vector
ffpry -l 5 chr2.fna > chr2.vector
ffpmerge chr*.vector > chrall.vector
```

The primary advantage of computing the vectors in this way is that separate chromosome vectors can be calculated on different processes and the process of *l -mer* counting can be distributed across a multi-processor architecture. If you have many small fasta files which you need to merge there is no clear advantages to using the count and merge process shown above. In this case, concatenating the files together on the fly is more effective:

```
cat *.fna | ffpry -l 5 > merged.vector
```

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1), ffprwn(1), ffpjsd(1), ffpcol(1)**

## NAME

ffpreprof − Constructs a relative entropy profile.

## SYNOPSIS

**ffpreprof [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

This shell script is a wrapper for the utility **ffpre**. Using it you can calculate the relative entropy over a range of feature lengths. Results are displayed in columnar format, where the length is the first column and the relative entropy is the second column. This script should be used to give an indication of the upper limit of *l* that is appropriate for phylogenic inference.

## OPTIONS

**−d, --disable-ry**
Use ATGC coding, default is RY.

**−a, --amino**
Input is amino acids.

**−t, --text**
Input is text.

**−r, --no-reverse**
Disable reverse complement matching.

**−s** *INT***, --start=***INT*
Specify start length range, default 3.

**−e** *INT***, --end=***INT*
Specify end length range, default 20. The maximum word length is 40. You may overide this maximum length by setting (and exporting) the environmental variable MAX_WORD_SIZE to a different value.

**−T** *DIR***, --tmpdir=***DIR*
Specify location for temporary files. Only applies for reading from standard in.

**−p** *DIR***, --path=***DIR*
Path to ffpre executable, if not in your path.

**−v, --version**
Print version information.

**−h, --help**
Print brief help message.

## EXAMPLES

To calculate a relative entropy profile using lengths between 4 and 12:

```
ffpreprof -s 4 -e 12 test*.fna
ffpreprof < test1.fna
```

For more information about the relative entropy measure, see **ffpre(1).**

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpvocab(1), ffpvprof(1), ffpre(1)**

## NAME

ffptree − Build Neighbor Joining or UPGMA trees from distance matrix input.

## SYNOPSIS

**ffpdf [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

Calculate a Neighbor Joining tree or UPGMA tree. This program is compatible with phylip style distance matrix infiles and the output produced by **ffpjsd** with option **-p.** The NEWICK style tree is written to standard out, while progress reports and a human readable tree are written to standard error. Input matrices can be read from standard input, a pipe or a file. The input file can contain multiple sets of matrices. FASTA sequences will be read from standard input if no file arguments are supplied and **ffptree** is called non-interactively (i.e. as part of a pipeline) or with a "-" in the argument list.

## OPTIONS

**−n, --upgma**
Build a UPGMA tree. The default is to build a neighbor joining tree.

**−l, --lower**
Use lower input triangle. Use only the lower half of the input matrix and assume symmetry.

**−u, --upper**
Use upper input triangle. Use only the upper half of the input matrix and assume symmetry.

**−t, --print-tree**
Disable printing of a human readable tree to standard error.

**−p, --progress**
Disable printing of tree build progress to standard error.

**−d, --print-data**
Enable printing of the input data before tree build standard error.

**−y, --symmetrize**
Symmetrize input data matrix, by averaging upper and lower triangles. By default, the input matrix is checked for symmetry, but is disabled with **−y.**

**−q, --quiet**
Disable printing of human readable tree, tree build progress and warnings to standard error.

**−o** *TAXAN*, **--outgroup=***TAXAN*
Rearrange tree to place taxon *TAXAN* at the outgroup position of the tree. *TAXAN* is an integer ranging from 1 to the number of input taxa.

**−O** *OUT*, **--out=***OUT*
Write Newick format tree to file *OUT.* By default, the tree is printed to standard out.

**−P** *OUT*, **--out-prg=***OUT*
Redirect tree build progress to a file *OUT.* By default, progress is printed to standard error.

**−w** *WDTH*, **--precision=***WDTH*
Specify the decimal precision used in reporting branch lengths in the Newick format tree. The default precision is 8 digits.

**−j[***S***], --jumble[=***S***]**
Jumble input order or species in matrix. Optional argument *S* can be used to provide a fixed random seed value to the random number generator, but the default value is the (system time) * (process ID).

**−m[***N***], --multiple[=***N***]**
Limit tree building to the first *N* sets. Without option *N* use all input sets, which is the default behavior. With option *N*, use up to 1 to the *N*th set. The default is to use all input sets from the input file without using option **−m.**

**−v, --version**
Display version information.

**−h, --help**
Display help message.

## EXAMPLES

This utility can be placed at the end of a pipeline of FFP utilities to directly generate tree output. The example below supressed printing of the build progress and the human-readable to standard error with the **−q option.**

```
ffpry -l 4 *.fasta | ffpcol | ffprwn |
ffpjsd -p species.txt | ffptree -q > treefile
```

Note, you must have defined a taxa name file with option **−p** in the call to **ffpjsd** to pipe input to **ffptree.** For additonal information see, **ffpjsd(1).** If you wish to view and save the build progress for later review, then redirect standard error to a file.

```
ffpry -l 4 *.fasta | ffpcol | ffprwn |
ffpjsd -p species.txt | ffptree > treefile 2> progress
```

Multiple input sets can be created and supplied in the same input file. Typically these will be pseudo-replicates generated via the **ffpboot** utility. If the file `input` contains 100 pseudo-replicates or input sets then by default **ffptree** will read all the input sets and generate trees for each. The **−m** option can be used to control how many input sets are processed.

```
ffptree < input # Generate all 100 trees
ffptree -m < input # Generate all 100 trees
ffptree -m50 < input # Generate first 50 trees.
```

Note the absence of an intervening space between and 50. This is the format for all options which take an optional argument.

## AUTHOR

This program was written by Gregory E. Sims

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpaa(1), ffpry(1),**

## NAME

ffpvprof − Constructs a word usage (vocabulary) profile.

## SYNOPSIS

**ffpvprof [***OPTION***] ... [***FILE***] ...**

## DESCRIPTION

This shell script is a wrapper for the utility **ffpvocab**. Using it you can calculate the word usage over a range of feature lengths. Results are displayed in columnar format, where the length is the first column and the word usage is the second column.

## OPTIONS

**−a, --amino**
Specify amino acid input. Default is nucleic acid (ATGC) input.

**−d, --disable-class**
Use ATGC coding. default is RY Coding. For Amino acid input this disables amino acid classes.

**−r, --no-reverse**
Disable reverse complement matching of nucleotide FASTA input. Does not apply for protein input.

**−z** *INT*, **--rand-mask=***INT*
Specify a random mask string. INT is the number of mismatches. The number of mismatches must be less than the default start value or less than the argument supplied to -s.

**−s** *INT*, **--start=***INT*
Specify start length range, default 1.

**−e** *INT*, **--end=***INT*
Specify end length range, default 20. The maximum word size is 40. You may overide this maximum length by setting (and exporting) the environmental variable MAX_WORD_SIZE to a different value.

**−f** *INT*, **--freq-thresh=***INT*
Specify word frequency threshold to count. Count words which occur only INT or more times. Default is 2.

**−T** *DIR*, **--tmpdir=***DIR*
Specify location for temporary files. Only applies for reading FASTA input from standard in.

**−p** *DIR*, **--path=***DIR*
Path to ffpry,ffpaa,ffpvocab executables, if not in path.

**−v, --version**
Print version information.

**−h, --help**
Print brief help message.

## EXAMPLES

To calculate a word usage profile using a frequency threshold of 3 for lengths between 4 and 12:

```
ffpvprof -s 4 -e 12 -f 3 vector
```

Using piping syntax, or as part of a pipeline of commands. The example below pipes input from standard in.

```
ffpvprof -s 4 -e 12 -f 3 < vector
```

## FURTHER DIRECTIONS

None.

## AUTHOR

This program was written by Gregory E. Sims.

## REPORTING BUGS

Report bugs to <gesims@lbl.gov>.

## COPYRIGHT



## SEE ALSO

**ffpvocab(1), ffpre(1), ffpreprof(1)**