



***Paracel Filtering Package™
User Manual***

Version 2.6

January 2002

1055 East Colorado Blvd.
Suite 5000
Pasadena, California 91106-2341
Phone: (626) 744-2000
Fax: (626) 744-2001
www.paracel.com

Copyright © 2002 Paracel Inc.

This document is the proprietary property of Paracel Inc., and is protected under federal copyright law, with all rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written consent of Paracel Inc.

Paracel Inc. provides this publication and software subject to the terms and conditions as defined in Paracel's *Agreement for Licensed Software*.

Paracel is a wholly-owned subsidiary of Applera Corporation through its business unit, Celera Genomics Group.

Paracel[®] is a registered trademark of Paracel Inc.

Paracel Filtering Package[™] is a registered trademarks of Paracel Inc.

Sun[®] is a registered trademark of Sun Microsystems, Inc.

Solaris[™] and SunOS[™] are trademarks of Sun Microsystems, Inc.

SPARC[®] is a registered trademark of SPARC International, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc.

Microsoft[®] Windows is a registered trademark of Microsoft Corporation.

All other trademarks are the sole property of their respective owners.

A special thanks to The Institute for Genomic Research for providing the sequence data used in the examples of this manual.

Contains IBM Runtime Environment for AIX[®], Java[™]2 Technology Edition Runtime Modules
Copyright © IBM Corporation 1999, All Rights Reserved."

Version 2.6 (01/2002) supersedes version 2.5 (10/2001).

Table of Contents

CHAPTER 1	<i>Introduction</i>
	Overview 1-1
	<i>Types of Contaminant and Repetitive Data</i> 1-2
	<i>Stages of Execution</i> 1-2
	<i>Debris and Statistical Output</i> 1-4
	Customer Support 1-6
CHAPTER 2	<i>Installation</i>
	Automatic System Installation 2-2
	Example Data 2-2
CHAPTER 3	<i>Command Line Usage and Arguments</i>
	Text Conventions 3-1
	Scylla Usage and Arguments 3-2

CHAPTER 4

PFPView

- Requirements 4-1
- Initiating PFPView 4-1
- Opening a File 4-2
- Display Section 4-3
 - Stage Section* 4-3
 - Hit Display Section* 4-3
 - Message Section* 4-5
 - Legend Section* 4-5
 - Sequences Panel* 4-6
 - Hits* 4-6
- Toolbar 4-8
 - File* 4-8
 - Stage* 4-9
 - Options* 4-9
 - Search* 4-13
 - Help* 4-13
- Closing PFPView 4-14

CHAPTER 5

Finding Repetitive and Contaminant Regions

- Atail 5-1
- DUST 5-2
- Minlen 5-2
- Maxlen 5-3
- PSEG 5-3
- Qualclean 5-4
- Trimjunk 5-4
- Searches 5-5
 - GMPROF – GeneMatcher profile comparison* 5-6
 - GMSW – GeneMatcher Smith-Waterman comparison* 5-6
 - HASTE – Hash-Accelerated Search Tool* 5-7
 - SW – Software Smith-Waterman comparison* 5-8

CHAPTER 6

Parameter Files

Editing Parameter Files 6-2

Parameter Parsing Rules 6-4

Parameter File Examples 6-5

APPENDIX A

File Management

The sequil Utility A-1

Examples A-3

rebaseupdate A-4

Examples A-5



The Parcel Filtering Package (PFP) is a full-featured sequence cleaning, filtering, and masking utility. PFP takes a single file and passes it through any number of user-defined stages to remove repetitive or contaminant elements from each sequence in the file. PFP's output file is a version of the input file that has been cleaned according to the user's specifications.

PFP consists of three executable modules, `scylla`, `sequtil` and `PFPView`. In addition, PFP contains sets of parameter and reference files for operating on several common organisms. `Scylla` is the executable that performs the filtering and masking functions. `Sequtil` is a sequence conversion utility that is provided to help manage PFP input and output files. `PFPView` is a graphical user interface that allows the user to analyse the PFP output file in detail. The parameter and reference files are used to customize processing to achieve optimal output.

Overview

The following sections describe types of contaminant data, stages of execution and statistical output.

Types of Contaminant and Repetitive Data

Contaminants are introduced into the sequence during laboratory processing. Repetitive regions are generally part of the sequence, but often cause problems during sequence analysis. The data that is filtered out of the sequence varies according to the user's specification, but usually includes such types of contaminants and repeats as: vector sequences, SINES, LINES and low complexity regions. There are specific data sets that can be used to filter these contaminants and repeats out.

- *E. coli* - Use NCBI's *E. coli* genome database, which can be found at: <http://www.ncbi.nlm.nih.gov/htbin-post/Taxonomy/wgetorg-?mode=Info&id=83333&lvl=3&keep=1&srchmode=1&unlock>.
- Mitochondrial / Chloroplast - Use NCBI's mitochondrial and chloroplast databases which can be found at: http://www.ncbi.nlm.nih.gov/PMGifs/Genomes/euk_o.html. Remember to match the species of the query sequence with the database chosen. If a database is not available for the species in question, try to find a database from another organism of the same genus.
- Vector - Use NCBI's UniVec database, which can be found at: <http://www.ncbi.nlm.nih.gov/VecScreen/VecScreen.html>. The UniVec database will also detect adaptors, linkers and PCR primers that may contaminate the query sequence.
- SINES and LINES - Short Interspersed Elements (SINES) and Long Interspersed Elements (LINES) can also confound sequence analysis and should be removed. SINES are usually 150-350 bp long and can be characterized as Alus (B1 in rodents) or Mammalian Interspersed Repeats (MIRs). LINES are usually complex, 1-6Kbp long, and can be characterized as L1 or retrotransposons. LINES can also be fragmented by other repeats and can therefore be hard to detect.
- Low complexity regions - Low complexity regions can be identified in a few different ways: using NCBI's SEG filtering program, using the DUST algorithm, or comparing the query sequence against known microsatellites for the species in question.

Stages of Execution

The input file for PFP is specified using the `-Query` flag. This file is then searched for repetitive or contaminant elements. The input file can be any of the following file formats: Paracel's Clustering and Assembly Markup Language (CAML), FASTA, IG, EMBL, phd, or GenBank. This file is incrementally passed through the user-defined

stages, each of which performs a cleaning action. The output of each stage is fed into the next. This means that changes in one stage affect later stages.

Note: In order to perform cleaning actions in the stages, the parameters required for each stage must be specified in a parameter file. Parameter files are discussed in detail in a subsequent chapter.

When the query file is passed through the various stages, each stage of execution consists of two distinct parts:

- Finding hits for undesired regions
- Using these hits to modify or annotate the input data

The user controls how each of these functions are performed.

There are several methods that PFP can use to find hits:

- Compare a query file to a set of reference sequences (e.g. RepBase)
- Compare a query file to a set of profiles
- Look for poly-A on the sequence 5' end and poly-T on the sequence 3' end
- Use DUST to find low-complexity regions in nucleotide sequences
- Use PSEG to find low-entropy regions of protein sequences
- Use cleaning to detect end regions of low quality
- Apply a minimum and maximum sequence length filter
- Look for end regions consisting primarily of characters from a specified set

Each of these methods will return a number of hits identifying regions within the query.

To annotate the repetitive or contaminant regions, the user can specify one "action" for each stage, using the `-Action` flag. The available actions are:

- Mask
- Filter
- Annotate (only for CAML input files)
- Excise

The Mask action specifies that PFP replace any hit regions in the query with an alternative, masking character. This masking character is specified with the

`-MaskChar` parameter. Then, the masking character is added to the scoring matrices to disallow matches with those regions in subsequent analysis.

The `Filter` action specifies that PFP only place sequences that did not match the contaminant or repetitive sequences into the output file. This has the effect of throwing out whole sequences and making the output file smaller than the input file. It is most useful for filtering sequences that are known to be entirely bad, e.g. an EST that had a high match threshold with *E. coli*.

The `Annotate` action applies only to CAML input files and specifies that PFP should not alter the sequence data. The output file will contain the same sequence data as the query file. Instead, PFP will add sequence annotations to each query sequence indicating the repetitive or contaminant regions that were found. The purpose of this function is to provide downstream applications the ability to choose whether or not to use the repetitive or contaminant regions. To enhance this ability, each annotation contains a special attribute that indicates what type of action downstream applications should take. The user can specify what goes into this attribute using the `-Annot` flag. The most common values are:

- `mask` - Mask the repetitive or contaminant region
- `excise` - Excise (remove) the repetitive or contaminant region
- `lowerq` - Lower the quality values associated with the repetitive or contaminant region
- `ignore` - Do not alter the repetitive or contaminant region

Note that for each stage where the `Annotate` action is specified, the user can also specify the `-Annot` flag to add a name to all annotations made at each stage.

The `Excise` action removes the hit regions from the query sequences. The hit regions are areas in the sequence that match repetitive or contaminant sequences. This action has the effect of shortening the query sequences and can be used in conjunction with `trimjunk` to clip off sequence end regions that are predominantly one character (e.g. the masking character). Using the `Excise` action may adversely affect the statistical output.

Debris and Statistical Output

In addition to the cleaned sequence output, PFP can optionally output other information about its execution.

The `-Debris` parameter is used to specify where to write all sequences that were filtered, as well as fragments that were masked or annotated. To keep track of which

sequences were filtered, use a unique file name for each filtering stage. At that point, the number of sequences in the Debris file plus the number of sequences in the output file will equal the number of input sequences.

The `-stat` parameter indicates where to write statistical details of the Cleanup procedure. In this file, PFP will list all matching regions for each stage and provide a summary of the prevalence of each repetitive or contaminant sequence. The user can use a different Stat file for each stage.

The statistics file contains two sections:

- **Hits found:** This section lists all the hits that PFP found for all stages. These hits are used to produce the statistical data in this file. The first line gives the labels for all fields in the ensuing lines. Each line lists the query name and base positions (indexed from 0), the reference name and base positions (indexed from 0), and the score of the hit. The reference name indicates what was found in that region of the query. For search stages, this will be a sequence name. DUST will attempt to identify the predominant bases in the masked region (e.g. AT-rich). The other algorithmic filters (`qualclean`, `minlen`, `trimjunk`, `atail`) simply list the algorithm name.

An example of this first section is:

QName	QStart -> QEnd	DName	DStart -> DEnd	Score
test_sequence+	70 -> 349	ALU+	0 -> 279	280
test_sequence+	644 -> 724	MIR-	20 -> 100	75
test_sequence+	1012 -> 1098	(AT)+	0 -> 86	45

- **Stage-specific statistics:** This section lists each stage in order. For each stage, PFP first lists the `-Alg` parameter that was used for that stage and the total number of bases input to that stage. Next, PFP lists one entry for each reference "sequence", indicating the number of hits each generated, the total number of bases covered by that reference sequence, and the percentage of the input covered by that reference sequence. After the listing of all the individual references, the numbers are summed and output. There are two sums that are output: a raw sum of the number of hits and the length (which is the exact sum of all the reference lines) and a corrected sum. The raw sum may include some bases twice if two hits have overlapped. The corrected sum accounts for this and only counts bases once. Both list the percentage of the query that was masked by this reference. An example of this section follows:

```

-----
Stage 1:  ref/alus.fasta
-----
Number of Query sequences in query/K03021.fasta:  1
Number of Query bases in query/K03021.fasta:    36595

Reference Seq Name      #Hits  Tot Len  Query%
      FLA                1         81    0.221%
      AluJo              13        1806    4.935%
      AluJb               4         526    1.437%
      AluSz               5        1126    3.077%
      AluSx               2         386    1.055%
      AluSg               3         495    1.353%
      AluSq               3         684    1.869%
      AluSp               5         924    2.525%
      AluSc               2         222    0.607%
      AluY                1         270    0.738%
      AluYa1              0          0    0.000%
      AluYb1              0          0    0.000%
      AluYa5              0          0    0.000%
      AluYb5              3         700    1.913%
      AluYa8              1         280    0.765%
      AluYb8              0          0    0.000%
      LTR26B              0          0    0.000%
      LTR26C              0          0    0.000%
      LTR26D              0          0    0.000%
-----
Total including overlaps  43        7500    20.495%
Total unique bases       43        6243    17.060%
-----

```

Customer Support

If you have any questions about the Paracel Filtering Package, contact Paracel Customer support for assistance.

- email: support@paracel.com
- Phone 9am-5pm PST

In the U.S.: 1-888-PARACEL

Outside the U.S.: 626-744-2000

- Phone 24 hours a day, 7 days a week: (626) 674-0707

The following chapter describes how to install the Parcel Filtering Package. Note that the release executables are compiled for the platforms described in Table 2-1. If one of the BioView[®] Toolkit (BTK) versions is selected, ensure that the PFP BTK version is the same version as the existing BTK installation.

TABLE 2-1. Binary Directories

Architecture	Binary Directory	Software Only	BTK 4.1.x	BTK 5.0.x
Solaris 2.6 / Sparc	sparc-sunos-5.6	Yes	Yes	Yes
Solaris 2.7 / x86	x86-sunos-5.7	Yes	Yes	Yes
OSF1 V4.0f / Compaq Alpha EV6	alphaev6-dec-osf4.0f	Yes	Yes	Yes
IRIX 6.5 / SGI MIPS	mips-irix-6.5	Yes	No	No
RedHat Linux 7.1 for x86	x86-redhat-7.1	Yes	No	Yes

Automatic System Installation

The PFP CD-ROM includes a script called `pfsetup.pl` that provides an interactive tool to perform the system installation. This script can be run by executing `perl pfsetup.pl`. The script will perform the installation steps that unarchive the files included with the CD-ROM.

For each user's environment, the `$PFPHOME` environment variable should be set to the Installation directory. For example:

```
setenv PFPHOME /paracel/paracel/pfp/CurrentRelease
```

During execution, `scylla` will look in this directory for the parameter, matrix, and reference files if they are not in the current working directory. This makes it possible to use `scylla` easily from a variety of directories without having to copy files or remember full paths. If you ever need to move the PFP installation, you can simply instruct the users to change their definition of `$PFPHOME`. The `bin.*` directories contain the executables compiled for the supported platforms and BTK versions. The appropriate directory for the user's platform should be in the user's path.

For example:

```
set path = ($path $PFPHOME/bin.sparc-sunos-5.6)
```

The BTK versions of this product require that the Oracle library (`libclntsh.so`) and HMMER library (`libhmmmer.so`) must also be in the `LD_LIBRARY_PATH`. These are included with the BTK distribution that comes with GeneMatcher™.

`libhmmmer.so` is typically in a directory like `~paracel/btk/lib` and `libclntsh.so` is in `$ORACLE_HOME/lib` which is typically in `/u01/app/oracle/product/8.0.5/lib`.

Once installation is complete, it is recommended that PFP be tested to make sure that it is operating correctly on the user's system. To do this, the user can run the example data, as described in the next section.

Example Data

The `$PFPHOME/example/` directory includes a single FASTA file, named `sample.fasta`, that can be used to test PFP. To run the example, copy `sample.fasta` into a writable directory and `cd` to that directory. Make sure that the `scylla` executable is in your path and that `$PFPHOME` is set correctly.

Execute the following command on a single line:

```
scylla -Query sample.fasta -Param pfp_human.prm \  
-Output masked.out -Stat stat.out
```

This command should take a few seconds to execute and should not produce any error messages. When it has finished, the masked query file will be saved to a file name `masked.out`. Statistics will be saved to `stat.out`.

Neither of these files should be empty.

In this example, the `scylla` command is directing the clean-up of the `sample.fasta` file using the parameters found in the parameter file `pfp_human.prm`. This is a standard set of parameters provided with PFP for masking human high-throughput genomic (HTG) data. Parameter files are discussed in more detail in a later chapter.

Command Line Usage and Arguments

The command line is used to control and customize PFP process including acquiring data, filtering and cleaning data, and producing the finished product. These arguments can also be placed into a parameter file, which will be described in detail in a subsequent chapter.

Text Conventions

- Required elements are shown without brackets. These elements must be specified on the command line or in a parameter file.
- Optional elements are shown with brackets “[]”.
- Commands will be shown in *courier*.
- Variables are shown in brackets < > and *italics*.
- All parameter names begin with a dash and are case-insensitive. In addition, the = sign between the parameter name and value is optional and is not shown in the usage. For example, `-Query=file` and `-query file` are equivalent ways for setting the query flag to be `file`.

Scylla Usage and Arguments

Usage

```
scylla
  [-Action <action>]
  -Alg <algorithm name>
  [-align]
  [-AlignOut <filename>]
  [-Annot <name>]
  [-Complement <Y|N>]
  [-DBHost <host_machine_name>]
  [-Debris <debris_filename>]
  [-GapExtn <gap_extn_penalty>]
  [-GapOpen <gap_open_penalty>]
  [-GMDir <GM_dir_name>]
  [-GMMountPoint <mtpt_name>]
  [-JunkChars <junk_char>]
  [-MaskChar <masking_char>]
  [-Match <match_score>]
  [-Matrix <filename>]
  [-Mismatch <mismatch_score>]
  [-NoEdits]
  [-NumProc <num_of_processors>]
  -Output <filename>
  [-Param <parameter_filename>]
  [-PFView <filename>]
  [-PolyDist <tail_length>]
  [-Priority <GM_priority>]
  -Query <query_filename>
  [-Reference <ref_lib_filename>]
  [-Stat <annot_stat_filename>]
  [-TempDir <temp_file_dir_name>]
  [-Threshold <score_threshold>]
  [-Verbosity <verbosity_level>]
  [-WordLen <word_length>]
  [-WProfThresh]
```

Arguments

`-Action` *<action>* (Optional)

Specifies the action to take for hits. Available actions are:

- `mask` – Masks the fragment from the query sequence using the character specified by the `-MaskChar` parameter. This is the default action.
- `excise` – Excises the hit region from the query sequence. The hit region is the area that matched a repetitive or contaminant sequence.
- `filter` – Removes an entire query sequence if it contains a hit with a reference sequence.
- `annot` – Specifies that the query sequence not be modified, instead the annotation is added to the CAML output file.

`-Alg` *<algorithm name>* (Required)

Specifies the search algorithm. Valid values are:

- `HASTE` – Hash-Accelerated Search Tool – (default)
- `SW` – Software Smith-Waterman comparison
- `GMPROF` – GeneMatcher profile comparison – specifies that GeneMatcher use a profile comparison to compare the query and reference sequences. The reference sequences must be in concatenated GCG profile format with individual profiles separated by lines containing only `///`.
- `GMSW` – GeneMatcher Smith-Waterman comparison
- `Atail` – Search for poly-A/T tail
- `DUST` – Search for low-complexity regions
- `Qualclean` – Search for low-quality sequence ends
- `Minlen` – Apply minimum length filter
- `Maxlen` – Apply maximum length filter
- `PSEG` – Identify low-entropy regions in protein sequences
- `Trimjunk` – Search for masked end regions

`-align` (Optional)

Specifies to perform banded Smith-Waterman alignments for each hit if the algorithm is HASTE. This can improve hit bounds at a small time penalty.

-AlignOut	<filename> (Optional)	Specifies the name of a file in which to write the alignments. This parameter can only be used if <code>-alg=GMSW</code> or <code>-alg=Haste</code> and <code>-align</code> are specified.
-Annot	<name> (Optional)	Specifies the name that should be used to annotate the changes to the query sequence throughout the CAML file. Examples of names are: <code>ignore</code> , <code>mask</code> , <code>excise</code> , <code>lowerq</code> . The default is <code>mask</code> .
-Complement	<Y N> (Optional)	Indicates whether to search both strands of the reference file of contaminant sequences specified by the <code>-Reference</code> parameter. If <code>atail</code> is specified by the <code>-Alg</code> parameter, and <code>-Complement</code> is specified, the algorithm will search for poly-A and poly-T tails. The <code>-Complement</code> parameter has no effect on the other algorithms that may be specified using the <code>-Alg</code> parameter. The default value is <code>Y</code> .
-DBHost	<host_machine_name> (Optional)	Specifies the host machine name for the BioView Server database. The default value for this option is to use the BioView Server setting.
-Debris	<debris_filename> (Optional)	Specifies the location to which discarded sequences and sequence regions are output.
-GapExtn	<gap_extn_penalty> (Optional)	Specifies the gap extension penalty for a search against a reference sequence file. It is overridden by any specification of <code>-matrix</code> . The default value is <code>-2</code> .
-GapOpen	<gap_open_penalty> (Optional)	Specifies the gap open penalty for a search against a reference sequence file. It is overridden by any specification of <code>-matrix</code> . The default value is <code>-8</code> .
-GMDir	<GM_dir_name> (Optional)	Specifies the directory under the mountpoint for database loading. This directory must already exist on the GeneMatcher. See the <i>ServerOne Software Manual</i> for details on creating directories on GeneMatcher.
-GMMountPoint	<mtpt_name> (Optional)	Specifies the database mountpoint for GeneMatcher search. For a detailed description of mountpoints, see the <i>ServerOne Software Manual</i> . The filename specified can be

- of the form *@filename* with each line in *filename* being a mountpoint. The default is to get the mountpoint from the `-DBHost` setting.
- `-JunkChars` *<junk_char>* (Optional)
Specifies the characters to ignore when `-Alg` is `minlen`, `maxlen` or `trimjunk`. The default is no characters.
- `-MaskChar` *<masking_char>* (Optional)
This parameter should not be changed unless special matrices have been constructed to score the new masking character properly. The default value is `X`.
- `-Match` *<match_score>* (Optional)
Specifies the match score for a search against a reference sequence file. It is overridden by any specification of `-matrix`. The default value is 1.
- `-Matrix` *<filename>* (Optional)
Specifies the matrix filename. The default value is none, meaning that no matrix is used. When this parameter is specified, it overrides the `Match`, `MisMatch`, `GapOpen` and `GapExtn` parameters.
- `-Mismatch` *<mismatch_score>* (Optional)
Specifies the mismatch score for a search against a reference sequence file. It is overridden by any specification of `-matrix`. The default value is -3.
- `-NoEdits` (Optional)
Applies only when the input file is in CAML format. If the parameter is specified, then any sequence edits in the original CAML file will be ignored during the processing of that stage.
- `-NumProc` *<num_of_processors>* (Optional)
Specifies the number of processors to use for the threading of the HASTE algorithm. The default value is 1.
- `-Output` *<filename>* (Required)
Specifies the file name for the annotated sequence output.
- `-Param` *<parameter_filename>* (Optional)
Specifies the parameter file. A parameter file contains a list of parameters and values that can be used during the PFP process. Commonly used parameters can be saved in

- a parameter file to avoid retyping them each time they are used. See the *Parameter Files* chapter of this manual for details.
- `-PFPView` `<filename>` (Optional)
Specifies the name of the output file that is compatible with PFPView. It is recommended that this file be named with a `.pfpv.caml` extension in order to differentiate it from other CAML files. See Chapter 4, *PFPView* for details on using PFPView.
- `-PolyDist` `<tail_length>` (Optional)
Specifies the tail length for an atail search (`-alg atail`). The default value is 40 bases.
- `-Priority` `<GM_priority>` (Optional)
Specifies the priority for a GM search. The default value is 0.
- `-Query` `<query_filename>` (Required)
Specifies the query sequence file.
- `-Reference` `<ref_lib_filename>` (Required if `-alg=haste|gmsw|gmprof`)
Specifies the reference library of contaminant or repetitive sequences to identify hits. This file can be in any of the following formats: FASTA, GenBank, EMBL, CAML, phd or IG.
- `-Stat` `<annot_stat_filename>` (Optional)
Specifies the filename for the annotation statistics output.
- `-TempDir` `<temp_file_dir_name>` (Optional)
Allows a directory to be specified in which to store all temporary files. This parameter is necessary because, in some cases, the temp files can be fairly large. If the system default temp directory is too small, use this option to re-route the temp file handling to a larger space. The default value is the current working directory.
- `-Threshold` `<score_threshold>` (Optional)
Threshold to use for the contaminant or repeat finding algorithm. The default value is 40.

-
- `-Verbosity` `<verbosity_level>` (Optional)
Defines how much information to print to the screen. A value of 0 is silent. A value of 1 will print progress messages for each batch of queries and the stages they go through. Higher values will print increasing amounts of information. Error messages are always reported. The default value is 1.
- `-WordLen` `<word_length>` (Optional)
Specifies the word length to be used for HASTE searches. The default value is 8.
- `-WProfThresh` (Optional)
Turns on weighted profile thresholds when `-Alg=GMPROF`. By default, the threshold used for profiles is based on the raw score returned by the profile search. For details, see Chapter 5, *Finding Repetitive and Contaminant Regions*.

PFPView is a graphical user interface that is used to visualize and analyze the results of the filtering process (see Figure 4-1). Matches to various undesirable sections of the query sequence are color coded and displayed with their corresponding information. Each hit can be analyzed in detail with the click of a mouse.

Requirements

PFPView requires approximately 40MB of RAM, with an additional 10MB of RAM required for each 10,000 sequences. For files with large numbers of sequences, PFPView may take several minutes to load the file.

PFPView is compatible with Windows 95, 98, NT and 2000, Dec Alpha OSF1 V4.0f, Solaris 5.6 and 5.7, and RedHat Linux 7.1.

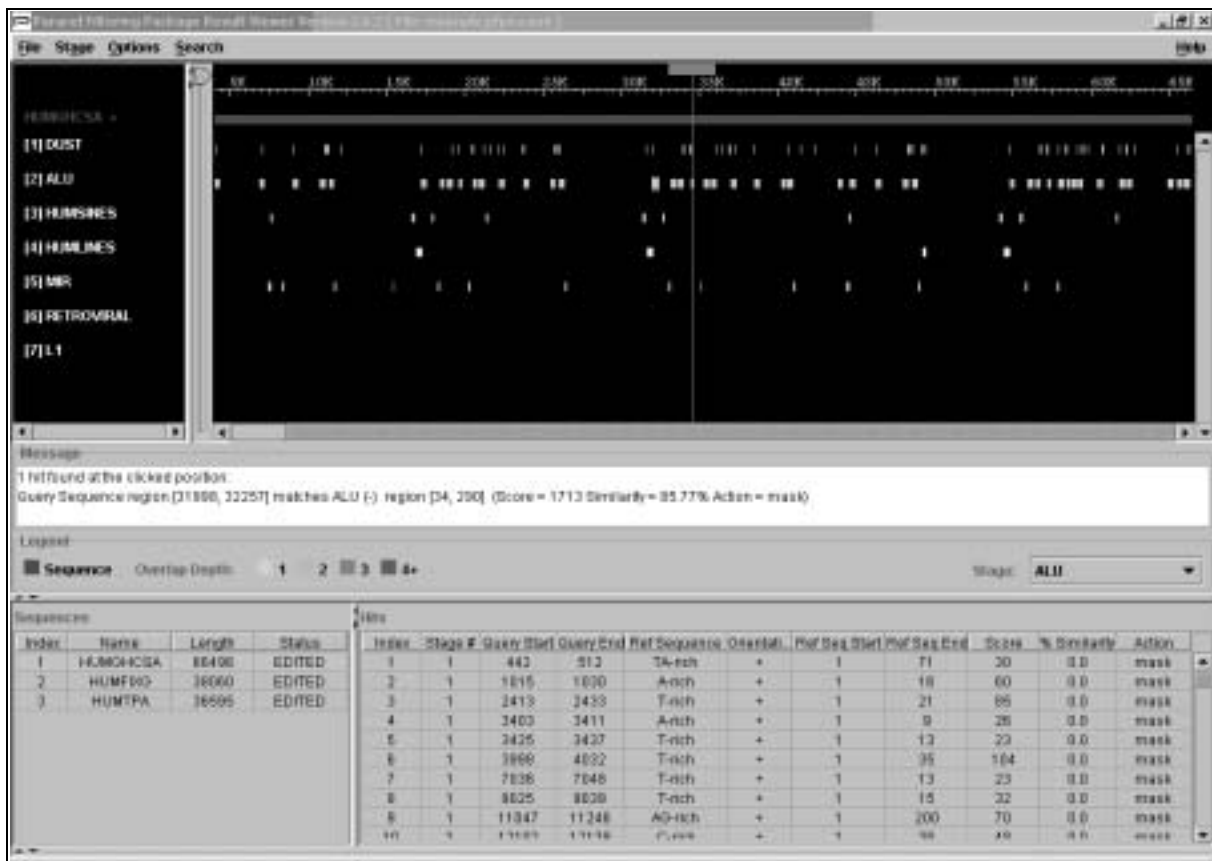
Initiating PFPView

In Windows, PFPView is initiated from the Start menu. In UNIX, PFPView is initiated by typing `pfpview` on the command line.

Opening a File

Once PFPView has been initiated, an Open File dialog box will be displayed. Use this dialog box to navigate to the desired filtering output file. Click on the output file and then on [OK]. Note that only CAML files that have been through the filtering process using Parcel Filtering Package (PFP), Parcel GenomeAssembler (PGA) or Parcel TranscriptAssembler (PTA) can be viewed. Other types of CAML files will create an error. In Windows, a DOS console will be launched first. This window will display processing and error messages. Next, the PFPView Java window will be opened with the output file displayed.

FIGURE 4-1: PFPView



Display Section

The Display section at the top of the PFPView consists of two sections. The Stage section and the Hit Display section.

Stage Section



The Stage section has the name of the sequence that has been filtered at the top of the panel and the names and index number of the various stages that were performed below it. The stages listed will vary according to those specified in the scylla parameter file. Each stage name has corresponding filtering information displayed to the right of it in the filtering Display section. Clicking on the stage name will highlight the name of that stage. Right clicking on a stage name will bring up a popup menu that provides two options: Hide Current Stage or Show All Stages. If the Hide Current Stage item is selected, the stage e.g. DUST, ALU etc., will be hidden in both the Stage and Display sections. To show all of the stages used during the filtering process, select the Show All Stages option.

Hit Display Section

Ruler

The yellow ruler at the top of the Hit Display Section indicates the base positions in the query sequence. By default, this ruler starts at 1. The increments will become smaller between ruler tick marks to match the scale of the sequence as the zoom is applied. The smallest increment, which can be seen at the highest zoom level, is one base. Using the zoom feature is described in a subsequent section.

FIGURE 4-2: Hit Display Section



Query Sequence and Filtering Stages

The query sequence is shown as a red line below the ruler. If the zoom is set to the highest magnification, the constituent bases are also shown. Below the query sequence, the hits that were found during processing are shown. Each hit represents a different kind of contaminant, area of low complexity, repeats, vector or poly-A/T. The types of hits depend on the kind of processing requested when filtering was performed.

Each stage that was performed is shown in the Stage section to the left of the hits produced. Each hit is color coded according to the stage in which it occurred and the number of hits at that location. To see an explanation of the color-coding scheme for a stage, click on that stage, and the color scheme will be shown in the legend section of the window.

On the left of the sequences and Hit Display area is the zoom control slider. To use the slider, place the cursor over the arrow, then click and move it up and down to the desired magnification. When the slider is at the top, a complete overview of the query sequence and all of the hits is shown. When the slider is at the bottom, or at maximum magnification, the individual bases in the query sequence will be shown; however the bases in the hits will not. To establish a reference point on which to focus zooming, use the focus locator that is displayed in the left upper corner when PFPView is initiated. To move the focus locator, place the cursor over the dark gray rectangle at the top. Click and drag the locator until it is centered over the desired area. The line that extends down from the rectangle will show the location in the query sequence and the hits from all stages. All zooming centers around the area selected.

To move from left to right in the Hit Display area, use the scrollbar at the bottom of the display area. To move vertically, use the scrollbar to the right of the display area. The hits may also be navigated by selecting a hit in the Display area and then clicking on the right or left arrow keys.

Right-clicking on a region with more than one hit pops up a menu of options: Highlight Longest Hit, Highlight Hits with Highest Score, Hide Current Stage and Show All Stages. To perform any of these actions, select the appropriate menu item.

Message Section

The Message section of PFView (see Figure 4-3) is used to display information about the query sequence, a stage or an individual hit. To see information about the query, click on the query name in the Stage display area. The Message area will show the number of bases. Click on the stage name in the stage section to show the total number of hits found by that filtering stage. Click on one of the hits found during any stage of filtering. These are the color-coded bars below the query sequence. The hit will be highlighted. The Message section shows the total number of hits, the position in the query sequence, and the name and region of the reference sequence that caused each hit at the clicked position. The region is specified in number of bases from the beginning of the query sequence and the reference sequence. The score of the hit and the percent similarity (PSIM) within each hit are also shown.

FIGURE 4-3: Message Section



Legend Section

The Legend section (see Figure 4-4), much like the Message section (see Figure 4-3), shows information about the query, stage or hit highlighted. The query sequence will always be shown in red. The color schemes for the hit overlap depths vary according to the stage referenced. Lighter colors indicate less hits and darker colors indicate more hits. The stage currently referenced can be changed by using the pulldown menu to the far right of the Message section.

FIGURE 4-4: Legend Section



Sequences Panel

The Sequences panel (see Figure 4-5) contains the following information in tabular format:

- **Index** – The order of the sequence in the query file, i. e. first=1, second=2, etc.
- **Name** – The name of the sequence.
- **Length** – The length of the query sequence in number of bases.
- **Status** – The post-processing status of the sequence. There are three possibilities:
 - **Untouched** – The sequence was not edited during processing.
 - **Edited** – The sequence was changed during processing
 - **Filtered** – The sequence was removed during processing. The manner in which the sequence was changed is dependant on the filtering parameters specified.

FIGURE 4-5: Sequence Panel

Sequences			
Index	Name	Length	Status
92	AC009483	206566	UNTOUCHED
93	AC009485	190706	UNTOUCHED
94	AC009486	203646	EDITED
95	AC009487	143050	UNTOUCHED
96	AC009488	203488	UNTOUCHED
97	AC009490	193657	UNTOUCHED
98	AC009491	174147	UNTOUCHED

Hits

The Hits panel (see Figure 4-6) contains the information of all hits of the currently selected sequence in tabular form:

- **Index** – The hit number in the order that it occurred.
- **Stage #** – The number of the stage in the order that it was performed.
- **Query Start** – The start position of the hit on the query sequence in number of bases. Indexing begins at 1.
- **Query End** – The end of the hit in number of bases. Indexing begins at 1.

- **Ref Sequence** – The reference sequence name and orientation. A plus sign indicates forward and a minus sign reverse orientation.

FIGURE 4-6: Hits Panel

Index	Stag...	Query St...	Query ...	Ref Sequence/Orient...	Ref Seq St...	Ref Seq ...	Score	% Simil...	Action	
55	1	86232	86254	A-rich	+	1	23	41	0.0	mask
56	2	152	237	ALU	+	135	219	327	70.93	mask
57	2	153	441	ALU	+	1	289	1944	88.58	mask
58	2	733	795	ALU	+	137	199	333	79.69	mask
59	2	734	1014	ALU	+	3	282	1875	80.81	mask
60	2	2439	2700	ALU	-	13	277	1890	89.81	mask
61	2	3438	3722	ALU	-	9	290	1785	84.56	mask
62	2	4033	4313	ALU	-	9	290	1773	84.17	mask

- **Orientation** - This column show the orientation of the hit: + for forward and - for reverse.
- **Ref Seq Start** – This is the number of bases into the reference sequence that the hit occurred. Numbering begins at 1 by default.
- **Ref Seq End** – This is the number of bases into the reference sequence that the hit ends. Numbering begins at 1 by default.
- **Score** – The score of the hit. The way the score is calculated will vary according to the algorithm selected to perform the comparisons.
- **% Similarity** – The percent similarity between the reference sequence and the query sequence in the area where the hit occurred. It may be set to 0.0 if the algorithm selected does not support percent similarity.
- **Action** – The value selected using the `-action` flag on the command line. Values include: Mask, Filter, Annotate or Excise.

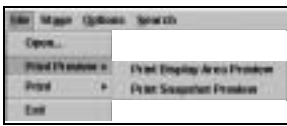
Note that clicking on any of the headings will sort the table according to the information in that column. Double-clicking any row in the Hits panel highlights the hit in the Display section. Information about the hit is also shown in the Message window and the Legend panel. Note that clicking on a hit in the Hits Display section will not highlight that hit in the Hits panel.

Toolbar

The toolbar has five menu options: File, Stage, Options, Search and Help. Each of these pulldown menus is described in a subsequent section.

File

The File pull-down menu has four options: Open, Print Preview, Print and Exit.



- **Open** – Opens another output file. If an output file is currently open, you will be prompted to close it. Note that only the special filtering CAML files produced by PFP can be displayed using PFPView. All other file types will produce an error.
- **Print Preview** – Offers two suboptions:
 - **Print Display Area Preview** – Previews a printout of the Display area. A separate popup window will be displayed showing the PFPView Display section. There are three available options in the toolbar of the popup window: (PRINT) to print the display area, (CLOSE) to close the popup window and a magnification pulldown menu. The magnification can be changed by selecting a value from the pulldown menu or by typing a value in the textbox.
 - **Print Snapshot Preview** – Previews a printout of the PFPView window. A separate popup window will be displayed showing the PFPView window as it would be printed. There are three available options in the toolbar of the popup window: (PRINT) to print the Display area, (CLOSE) to close the popup window and a magnification pulldown menu. The magnification can be changed by selecting a value from the pulldown menu or by typing a value in the textbox.
- **Print** – Offers two suboptions:
 - **Print Display Area** – Prints the display of the filtering process
 - **Print Snapshot** – Prints the entire PFPView window
- **Exit** – Closes PFPView

Stage



This pulldown menu shows all the stages of filtering performed on the query sequence. By default, all of the stages that were performed are selected, meaning that they are also shown in the Display area. To hide a stage that was performed, deselect it. Deselected stages will not appear in either the Stages area or the Display panel. The stages are listed in the order they were performed.

Options



There are several choices in the pulldown menu for this Toolbar option. They are: Show Reference Stage Statistics, Show Reference Sequence Statistics, Show Parameters and Advanced.

Show Reference Stage Statistics

To show the reference stage statistics, go to the Toolbar and click on Options. In the pulldown menu, click on the checkbox next to "Show Reference Stage Statistics". The reference stage statistics will be shown in a separate panel below the Hit and Sequence panels. There are several columns. These are: Index, Ref Stage, # of Bases, # of Unique Bases, # of Hits, % of Hit, and % of Unique Hit.

- **Index** – The number of the stage in the processing.
- **Ref Stage** – The stage of the filtering process. This will vary according to the stages requested on the command line.

FIGURE 4-7: Reference Stage Statistics Panel

Index	Ref Stage	# of Bases	# of Unique Bases	# of Hits	% of Hit	% of Unique Hit
2	ALU	1071480	1571242	6422	15.081	6.745
3	HUMSINES	1281225	1125854	6536	7.289	6.285
4	HUMLINES	3480502	2888471	8829	19.426	15.170
5	MIR	633888	574387	8838	3.088	2.887

- **# of Bases** – The sum of all of the query sequence bases involved in a hit. Note that this number counts each query base in each hit and sums all of them. Consequently, this number may be counting a single query base multiple times if there are overlapping hits.
- **# of Unique Bases** – The actual number of query bases involved in the hits. This is the corrected sum from the # of Bases value described above. In other words, each query base involved is counted only once.

- **# of Hits** – This column displays the number of hits generated by the stage.
- **% of Hit** – The percentage of the query covered by the # of Bases column.
- **% of Unique Hit** – The percentage of the query covered by the # of Unique Bases column.

Show Reference Sequence Statistics

To show the reference sequence statistics, first select the Show Reference Stage Statistics option from the Options pull-down menu on the toolbar. Select the toolbar option again and select the Show Reference Sequence Statistics option. The Reference Sequence Statistics will be displayed in the lower panel, see Figure 4-8. When the Reference Sequence Statistics are first displayed, the column headings are shown, but no data are displayed. Select a stage from the list in the Reference Stage Statistics panel. The corresponding data will be shown in the Reference Sequence Statistics section. Note: To select a range of stages from the Reference Stage statistics, hold down the Shift key and click on the sequence at the start of the range and then click on the stage at the end of the range. To select individual stages, hold down the Ctrl key and select the desired stages by clicking on them. There are several columns of data in the Reference Sequence Statistics panel. They are: Index, Ref Sequence, # of Bases, # of Hits and % of Hit.

- **Index** – The stage index number. Use this number to find the corresponding stage name in the Reference Stage Statistics panel.
- **Ref Sequence** – The name of the reference sequence that generated the hit.
- **# of Bases** – The number of bases in the query sequence that are involved in hits with this reference sequence.

FIGURE 4-8: Reference Sequence Statistics

Index	Ref Sequence	# of Bases	# of Hits	% of Hit
1	FAM#SINE(A)u8#0D;	2239	27	0.012
2	FLAM_A#SINE(A)u8#0D;	866	16	0.0050
3	FLAM_C#SINE(A)u8#0...	785	10	0.0040
4	FRAM#SINE(A)u8#0D;	1192	13	0.0070

- **# of Hits** – This is the number of hits generated by the reference sequence.
- **% of Hits** – This is the percentage of the query covered by hits with this reference sequence.

Show Message Panel

This option, if set, shows the Message Panel. To hide the Message Panel, deselect this option.

Show Legend Panel

This option, if set, shows the Legend Panel. To hide the Legend Panel, deselect this option.

Show Parameters

To show the parameters used to create the output currently displayed, click on the Show Parameters checkbox. There is a tab at the top of this box for each stage specified in the parameter file. These will vary according to the stages requested. All of the parameters selected for each stage will be shown. The global tab will display the output file directory name, the output filename and the query sequence filename. The parameters available for each stage are those previously described for use on the command line. If an option was not specified on the command line, either its default is displayed or the parameter is left blank. When you have finished browsing through the parameters, click on (CLOSE), and the Parameters Box will disappear.

FIGURE 4-9: Parameters Box



Show Quality Values

Select this option to view the average quality value of the base calls in a sliding window of user-specified width. The width of the window can range from a single

base to 200. This option is enabled only when the input data file contains QV data. To determine the index number of a base, click at the desired position on the plot. The position will be indicated by a vertical yellow line, and the index number of the position will be displayed next to the line. This can be done for multiple bases. To delete the position lines and index numbers, right-click anywhere in the Quality Value display.

Show GC Content

Select this option to view the GC content of the query sequence. The plot represents the % GC content of the bases in the sliding window. The size of the sliding window, which ranges from 20 to 500 bases in width, can be changed using the slider to the left of the plot. You can determine the index number of a base in the GC Content display in the same way as you do in the Quality Value display.

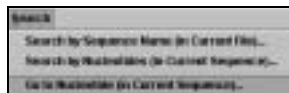
Advanced

This option has only one suboption, Index Nucleotides from Zero which re-indexes the nucleotides in the query and reference sequences from zero. Default indexing is from one.

Search

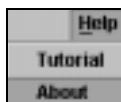
The Search Toolbar option has three selections: Search by Sequence Name, Search by Nucleotide and Go to Nucleotide. Each of these options can be used to filter the available information or to find a specific location in the query sequence.

- **Search by Sequence Name** – This option, if selected, will bring up a popup window. In the textbox, type the sequence name to search for in the current output file. If an exact match is required, leave this checkbox selected. If an inexact match is required, this box should be deselected.
- **Search by Nucleotide** – To search for a specific nucleotide pattern in the currently selected sequence, enter the desired pattern in the Query String textbox. Note that cutting and pasting is not allowed and that the search will be case insensitive. To specify the area of the query sequence to be searched, type the nucleotide number into the “From Base Index” textbox. By default query sequences are searched from 1. Query sequences will be searched from 0 if the Index Nucleotides from Zero option is set. Only the part of the query that occurs after the nucleotide specified will be searched. To leave the dialog box open, deselect the “Close this dialog when search is done” checkbox. Otherwise, leave it selected. When all of your parameters have been set, click on (OK). The first sequence that matches the pattern specified will be displayed in the Sequences panel at the bot-



tom of the screen. If no sequences matched the specified pattern, a dialog box will be displayed stating "Search finished. Query string not found." To acknowledge this, click on (OK).

- **Go to Nucleotide** - To move to a specific nucleotide in the query sequence currently displayed, select the "Go to nucleotide" option. By default, the query sequence is indexed from 1. If the Index Nucleotides from Zero option is selected, the query sequence will be indexed from zero. A dialog box will be displayed with the current sequence name and the current sequence length shown. Enter the position number of the desired nucleotide in the "Go to base index" textbox. Click on (OK). The zoom will be set to maximum and the nucleotide specified will be shown. This function is useful for inspecting specific hits listed in the Hits panel.



Help

The Help toolbar option has two options: Tutorial and About.

- **Tutorial** – Displays a brief tutorial of PFPView.
- **About** – Displays the software name, version number and copyright information.

Closing PFPView

To close PFPView:

1. Select Exit from the File Menu on the toolbar.
2. Click on the X in the right upper corner of the DOS window.

PFPView should close gracefully. *Do not close the DOS window first, as this will cause an error.*

Finding Repetitive and Contaminant Regions

This chapter discusses the different methods used to find contaminant regions in the query file. The user can choose between these methods by using the `-Alg` parameter.

Important Note: The value of the `-Alg` parameter is case insensitive.

Atail

This algorithm, specified using `-Alg atail`, offers an efficient way to identify poly-A repeats on a sequence's 3' end. If `-Complement Y` is specified, `atail` will also look for poly-T repeats on the 5' end. The `-PolyDist=<int>` parameter specifies how long of a region to examine for each sequence. The `-Threshold=<int>` parameter specifies the score threshold for returning a hit. When searching for the poly-A/T, `atail` will use the substitution scores specified by the parameters `-Matrix`, `-Match` or `-Mismatch`.

DUST

This low-complexity annotation algorithm is invoked by using `-Alg dust`. The algorithm is based on the `SIMPLE34` algorithm described in “SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences” by John M. Hancock and John S. Armstrong, *CABIOS Communications*, v10 n1, 1994 pp 67-70. This information is also available at <http://life.anu.edu.au>.

The algorithm identifies regions of low-complexity, where tri-nucleotide motifs are repeated at a higher than expected rate. The threshold controlling this algorithm is specified using the `-Threshold` parameter, where any region scoring greater than or equal to `-Threshold` will be considered low-complexity. We have empirically determined that a value of 22 yields promising results.

Minlen

This is a type of annotation that allows you to identify sequences based on length characteristics. By specifying `-Alg minlen` for a given stage, the user is specifying that each sequence in the query file be examined, creating a full-length hit for each sequence that does not pass the user's length criteria.

The criteria are specified by the `-Threshold` and `-JunkChars` parameters. A sequence passes the criteria and does not generate a hit if the number of ‘good’ bases in the sequence is greater than or equal to the value of the `Threshold` parameter. A “good” base is any base not specified in `JunkChars`. Thus if `JunkChars` is empty or not specified, then `Threshold` is measured against the total sequence length. If `JunkChars` is `NX`, for example, the effective sequence length is the number of characters in the sequence that are neither `N` nor `X`.

If the number of good bases is below the `Threshold`, a hit is generated that covers the full length of the query. The `minlen` type of `-Alg` is meant to be used as a filter (i.e. with `-Action filter`) and will generate a parameters warning message if this is not the case. It can be used, however, with a `mask` or `annot` action if the user does want to `mask` or `annotate` the entire sequence length.

Maxlen

This algorithm works in a similar manner to `minlen`. This is a type of annotation that allows the user to identify sequences based on length characteristics. By specifying `-Alg maxlen` for a given stage, the user is specifying that each sequence in the query file be examined, creating a full-length hit for each sequence that does not pass the user's length criteria.

The criteria are specified by the `-Threshold` and `-JunkChars` parameters. A sequence passes the criteria and does not generate a hit if the number of "good" bases in the sequence is less than or equal to the value of the `Threshold` parameter. The `-Threshold` parameter specifies the maximum allowable sequence length. For example, if the threshold is 1,000, then all of the sequences that are longer than 1,000 bases will generate a hit over the full sequence length. A "good" base is any base not specified in `JunkChars`. Thus if `JunkChars` is empty or not specified, then `Threshold` is measured against the total sequence length. If `JunkChars` is `NX`, for example, then the effective sequence length is the number of characters in the sequence that are neither `N` nor `X`.

If the number of good bases is above the `Threshold`, then a hit that covers the full length of the query is generated. The `maxlen` type of `Alg` is meant to be used as a filter (i.e. with `-Action filter`) and will generate a parameters warning message if this is not the case. It can be used, however, with a `mask` or `annot` action if the user does indeed want to mask or annotate the entire sequence length.

PSEG

The PSEG algorithm is specified by setting the `-Alg` parameter to `pseg`. It was specifically developed to identify low-complexity regions in protein sequences using an entropy measure. It breaks the sequence down into overlapping windows, the size of which is specified with the `-PsegWin` parameter, and calculates an entropy score for each window. Within each window, it looks for a single-position entropy of less than `-PsegLowCut`. This nucleates a further region examination in which PSEG looks to the left and right of that position to find the largest local low-entropy region. This local region must have entropy less than `-PsegHighCut` at all positions. Adjacent high-entropy, `> -PsegHighCut`, regions may be combined with the low-entropy regions if the length of the high-entropy region is less than `-PsegLenMin`.

See 'Statistics of Local Complexity in Amino Acid Sequences and Sequence Databases,' by John C. Wootton and Scot Federhen, *Computers Chem.* V 17, No 2, pp 149-163, 1993. for details on this algorithm.

Qualclean

The quality value cleaning method is specified by using `-Alg qualclean` as an argument. This algorithm looks at the ends of each sequence to detect the low-quality clipping bounds. The only threshold for this method is `-Threshold=<int>` which indicates the threshold for bad quality data (any data less than or equal to the value of `-Threshold` will be considered poor for the purpose of the algorithm). The algorithm starts at the beginning and end of the sequence, iterating inwards as long as the average quality of the data over the last 30 bases is less than `-Threshold`. Hits generated by the quality value cleaning method will always be anchored to either the start or end of a sequence.

Trimjunk

The TRIMJUNK algorithm allows the user to identify beginning and end regions of sequences that consist mainly of a specified set of characters. This could be used, for example, to identify end regions that consist predominately of 'N' and 'X' residues. The set of "junk" characters is specified via the `-JunkChars` parameter.

The TRIMJUNK algorithm uses the `-Threshold` parameter as a setting for what percentage of the bases are junk characters. For example, if `-Threshold` is set to 95 and `-junkchars` is set to "NX", TRIMJUNK will look for regions that are composed of 95% or more of N or X characters.

The algorithm first examines the 5' end of the sequence, iterating forward through the sequence and finding the longest region anchored at the 5' end of the sequence that matches the threshold and junk characters criteria, if any. This step is repeated for the 3' end of the sequence, iterating in reverse through the sequence and finding the longest region anchored at the 3' end. The iteration inward will stop early if the TRIMJUNK algorithm finds a 40 base region that is below the `Threshold`. If the identified 5' and 3' regions overlap, they are combined into a single region stretching the length of the sequence. Otherwise, the regions are reported independently as hits if they have non-zero length.

Searches

The other algorithms for detecting repetitive or contaminant regions perform a search between the query file and a reference library of sequences. This reference library of sequences is specified via the `-Reference=<file>` parameter. This reference library is typically a RepBase file, mitochondrial data, *E. coli* contaminants, or cloning vector contaminants. The search between the two files returns hits indicating the regions of similarity. These hits are then handled per the `-Action` specification.

There are many parameters for the search. The following parameters apply to all search types:

- `-Threshold` – This parameter specifies the score threshold for the search.
- `-Matrix` – This parameter specifies the scoring matrix to use for searching. The matrix must be in FASTA format. If this is specified, then the values in the matrix override the `Match`, `Mismatch`, `GapOpen` and `GapExtn` options.
- `-Match` – This parameter specifies the match score for the search. This must be a positive number.
- `-Mismatch` – This parameter specifies the mismatch score for the search. This value must be a negative number.
- `-GapOpen` – This parameter specifies the gap open penalty. This value must be a negative number.
- `-GapExtn` – This parameter specifies the gap extension penalty. This value must be a negative number.
- `-Complement` – This parameter specifies that both orientations of the reference file should be searched. *Note:* use `-Complement=N` when the reference file is already in both orientations.
- You can specify what search type to use for comparing the query and reference files. This is done by using the `-Alg` parameter. The legal values for `<algorithm_name>` are:
 - `GMPROF` – GeneMatcher profile comparison
 - `GMSW` – GeneMatcher Smith-Waterman comparison
 - `HASTE` – Hash-Accelerated Search Tool
 - `SW` – Software Smith-Waterman comparison

The following is a more detailed description of each of these algorithms and their parameters.

GMPROF – GeneMatcher profile comparison

This search type uses a GeneMatcher to perform a profile search to compare the query and reference sequences. The reference sequences must be in concatenated GCG profile format, with the individual profiles separated by lines containing only “//”. The GMPROF algorithm uses multiple high-scoring pairs to identify regions within the query, and is therefore appropriate for both masking and filtering applications.

By default, the threshold used for profiles is based on the raw score returned by the profile search. With the `-WProfThresh` option turned on, the threshold is a weighted threshold that has been scaled to the range [0, 1] based on the profile length and maximum possible score. A raw profile score (as returned by BTK) can be converted to a weighted score following this formula:

$$S_w = \text{SQRT}(S_r/S_{\text{max}} * L_h/L_p)$$

where:

S_w : Weighted score

S_r : Raw profile score

S_{max} : Maximum score possible for that profile

L_h : Length of the hit (w.r.t. profile)

L_p : Length of the profile

To turn on the use of weighted scores, you must specify the `-WProfThresh` flag. When this is specified, then the value of `-Threshold` is equivalent to $S_w * 1000$. The scaling of 1000 is used so that `-Threshold` can remain an integer. So, for example, if you specify `-WProfThresh -Threshold 850`, this is equivalent to using a weighted threshold T_w of 0.85 (so that only scores S_w of 0.85 or more are returned).

GMSW – GeneMatcher Smith-Waterman comparison

This search type uses a GeneMatcher to perform a linear or affine (matrix-dependent) Smith-Waterman comparison. This is the optimal local comparison between the query and database files. This algorithm is useful for both masking and filtering because it offers Multiple High Scoring Pairs (MHSP). This is the optimal algorithm to use for highly-sensitive masking. The options for this algorithm are described in subsequent paragraphs.

GeneMatcher Configuration for GMSW and GMPROF

By default, GMSW and GMPROF will run on the GeneMatcher available in the BioView Server setting. This is typically indicated with the `-DBHost` parameter, which specifies the host machine for the BTK database. To run on a different GeneMatcher you must specify the full GeneMatcher mountpoint using the `-GMMountPoint= <name>` parameter as well as the BTK database host to which it is attached using the `-DBHost=<name>` parameter.

To specify a single GeneMatcher, `<name>` must be in the format `/fdF/GMNAME/gmX/Y`. To specify multiple GeneMatchers, you can specify `-GMMountPoint= @FILENAME`, where each line in `FILENAME` contains a mountpoint. To specify a directory on the GeneMatcher (under `gmX/Y`) you must use the separate `-GMDir=<name>` option. For example, to use the GM directory `/fdF/genematcher/gm0/2/TEST`, specify `"-GMMountPoint=/fdF/genematcher/gm0/2 -GMDir=TEST"`. The rationale behind separating the `GMMountPoint` and `GMDir` arguments is to allow the use of multiple GeneMatchers, where the same directory appears on each GeneMatcher (as is required by BTK) but the mountpoint and hard drive configurations may differ.

If a configuration where the BTK Oracle host is not located on the local machine is being used, you may have to use the `-DBHost` parameter to set the machine name of the BioView Server host.

The `-Priority` parameter lets you set the GeneMatcher search and post-processing priority for the search. Valid range is -99 to 99, with a higher number having a higher priority. Prioritization is handled by BTK.

HASTE – Hash-Accelerated Search Tool

This is a software hash search algorithm. Hash word length can be set via the `-WordLen` parameter. The algorithm uses an initial hash lookup, followed by a gapless hit extension. The algorithm then attempts chaining on the gapless hits to approximate the Smith-Waterman alignment. The smaller the value for `-WordLen`, the more sensitive and slower HASTE will be. Usually a word length of 8 is sufficient for finding most repeats, although a word length of 4-6 is helpful for finding the more distant matches. The `-align` flag turns on post-processing banded Smith-Waterman alignments that may increase the accuracy of the HASTE hit bounds at a small time penalty. This is helpful for maintaining a high level of accuracy in the masking.

The HASTE algorithm is the appropriate software algorithm to use for typical masking and annotation because it is fast and it can generate multiple hits per query-database document pair (MHSP).

Note: To run HASTE using multiple processors (multi-threading), use the `-NumProc` option.

SW – Software Smith-Waterman comparison

This is a software implementation of the optimal Smith-Waterman local alignment algorithm. This is the slowest algorithm and is not recommended for general usage. It can be useful in verifying or comparing results on small datasets.

PFP can be run solely from values passed in on the command-line. The program also offers the ability to use a separate parameter file for filtering and masking. The difference between these choices is that the command-line is limited to a single stage, meaning that the user can only specify one stage of execution. To use the functionality of multiple stages (and the automatic piping of the data through these stages), the user must indicate these stages in a separate parameter file.

PFP includes several parameter files that have been configured for various organisms. These parameter files reside in the `$PFPHOME/param` directory. Assuming that the user's `$PFPHOME` environment variable has been set correctly, the user can access these parameter files by typing the file name. Scylla will search under `$PFPHOME` for the specified parameter file.

The following parameter files are included in PFP for masking high-throughput genomic (HTG) data. These parameter files are set to find for low-complexity regions using DUST and organism-specific repetitive elements using HASTE.

- `pfp_aratha.prm` - Masking *Arabidopsis thaliana* (wild mustard plant)
- `pfp_celeg.prm` - Masking *Caenorhabditis elegans* (nematode)
- `pfp_danr.prm` - Masking *Danio rerio* (zebrafish)
- `pfp_dros.prm` - Masking *Drosophila melanogaster* (fruit fly)
- `pfp_human.prm` - Masking *Homo sapiens* (human)

- `pfp_mouse.prm` - Masking *Mus musculus/domesticus* (mouse)
- `pfp_rat.prm` - Masking *Rattus norvegicus* (rat)

There is a second set of HTG parameter files that perform the same task for the same organisms, but add a GeneMatcher search step in order to maintain the highest level of accuracy. These are named similarly, with the 'pfp' replaced by 'pfpgm', e.g.: `pfpgm_aratha.prm`, `pfpgm_celeg.prm`, etc.

There is a third set of parameter files for the filtering of Expressed Sequence Tags (EST) data. In addition to masking repetitive elements, these parameter files mask vector contamination and filter sequences with high similarity matches to mitochondria, RNA or *E. coli*. Short sequences are also removed. An example filename for these parameter files is: `pfpest_aratha.prm`.

As mentioned previously, to use these parameter files the `$PFPHOME` variable should be set up correctly and the file name specified. For example, to mask the human chromosome 22 (`chr22.fasta`) the user can go to the directory in which `chr22.fasta` resides and use a command similar to the following:

```
scylla -Query=chr22.fasta -Param=pfp_human.prm \  
-Output=chr22.masked -Stat=chr22.stat
```

This command will have the effect of masking `chr22.fasta` using the parameters specified in `$PFPHOME/param/pfp_human.prm`. The masked output is sent to a file named `chr22.masked` and statistical output is written to `chr22.stat`.

Editing Parameter Files

If the included parameter files are insufficient for the filtering and masking to be performed, parameter files can be created for use in PFP. The easiest way of doing this is to copy an existing parameter file and modifying the copy to perform the masking and filtering that the user desires. The parameter files in PFP are stored in the `$PFPHOME/param/` directory and have the names listed previously. The user can copy one of these files to a working directory and then modify it to fit the needs of the data. If this new parameter file is useful to a wider group of people, it may be beneficial to rename it and have a system administrator place it into the `$PFPHOME/param/` directory for others to use. Note that if the modified parameter file is not in the `$PFPHOME/param/` directory, the user will have to specify the full path to the parameter file in order to use it.

This section and the following section are intended to help the user read and modify parameter files.

The format of the parameter file is fairly simple and free-form. Within the parameter file the user specifies parameters in the same manner as on the command-line. Consecutive whitespace and '=' signs are treated as a single character of whitespace. Inside the parameter file, the user specifies the stages of execution by using a special % tag followed by a one-word description of the stage. Note that this description cannot contain any whitespace. Each subsequent % tag and one-word description indicates the information for a new stage. This is best illustrated with an example. Note that comments begin with a '#' and continue to the end of the line.

```
# This is a sample parameters file
-MaskChar      X
-Complement    Y
-Alg           Haste
-Annot         mask
-align

%ATAIL
-Reference     atail
-PolyDist      40
-Threshold     8
-Action        mask
-Debris        /dev/null

%DUST
-Reference     dust
-Threshold     22
-Action        annot
-Debris        /dev/null

%VECTOR
-Reference     reference/UniVec
-Threshold     30
-Action        mask
-WordLen      8
-Matrix        matrix/dna.plm5.1.mat
-Debris        /dev/null
```

Think of the parameter file in two main sections: The general parameter section, and the stages section. Every parameter specified before the first % tag is a "general" parameter--it applies to every stage. In the previous example, these general parameters begin with `-MaskChar X` and end with `-align`. These parameters will apply to all stages by default. These general parameters can then be specifically overridden by individual stages that specify the same parameter.

Parameter files are specified via the `-Param` parameter on the PFP command line. When this is done, PFP will parse the specified files along with the other command line options in order to determine how to execute. The rules for how PFP interprets its command line and parameter file options are complex to allow for maximum flexibility.

Parameter Parsing Rules

In its most general form, PFP allows multiple options and multiple parameter files to be specified on the command-line.

- The parameter files cannot be recursive--that is, a parameter file cannot specify another parameter file.
- Stages can only be specified within a parameter file.
- There are several levels of 'defaults' that PFP allows. At the most basic level, PFP has internal defaults for many of its parameters. These are what PFP displays in its usage message (type 'scylla' and hit RETURN). There are not internal defaults for all the parameters because there are some things PFP cannot guess (e.g. query file, reference files). The internal defaults are overridden by any new specification of that parameter on the command line or in the parameter file.
- When PFP parses the command-line, it does so in three passes. As it makes these three passes, it builds an internal array of parameters to use for each stage. The parameters in the array are overwritten as new, overriding values are parsed.

In the first pass, PFP reads all command-line parameters that begin with '--'. For example, --Threshold=20 or --Alg=Haste. These are known as the user defaults because they are the user's way of specifying default parameters that can be overridden by other parameters or parameter files. These user defaults override PFP's internal defaults and form the basic set of parameters that is used as a template for each stage.

In the second pass, PFP parses all parameter files specified with the `-Param=<file>` argument. These files are parsed in order (reading from left to right in the command-line). PFP begins to parse these files in default-override mode. This mode continues as long as the parameter files being parsed do not contain % tags. In this mode, any parameter in the file will modify the user defaults for all stages.

Once a file is encountered that has a % tag, PFP leaves default-override mode and enters stage mode. Each subsequent parameter file is then treated as outlined above, with the general parameters overriding the user defaults and the stage parameters overriding the general parameters. Any new stage that is encountered is appended to the current list of stages. This continues until the last parameter file has been parsed.

The final pass to parse the command-line takes care of all command-line parameters that begin with '-'. For example,

```
-GMMountPoint=/fdf/genematcher/gm0/0, -Priority=90
```

These are known as global override parameters. Now that PFP has parsed all the stage information from all the parameter files, the program now applies these global override parameters to modify all stages. The values specified in this manner will become the values that are used for all stages, regardless of what was in the parameter files.

Parameter File Examples

Example 1: Basic FASTA File Masking

Given the query file `test.fasta`, we want to look for all occurrences of the Alu repeat and mask them (replace the regions with X).

Our input file, `test.fasta`, is:

```
>test_sequence
agcgactgacgtagecgcctatgagctgagcgtagctgagctgagctgacgagccgcgcccccca
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcggggcgatcacctgagg
tcaggagttcgagaccagcctggccaacatggtgaaaccccgctctactaaaaatacaaaaattagccg
ggcgtggtggcgcgccctgtaatcccagcactcgggaggctgaggcaggagaatcgcttgaacccggg
aggcggaggttgcagtgagccgagatcgcgccactgcactccagcctgggacagagcgagactccgctc
ggcggccggtatataatagcgcgcccgatgctacgatcgatcgatgctgacatcgactcgactcgcgac
```

Our reference file, `alu.fasta`, is:

```
>Alu
ggccgggcgcggtggctcacgcctgtaatcccagcactttgggaggccgaggcggggcgatcacctgagg
tcaggagttcgagaccagcctggccaacatggtgaaaccccgctctactaaaaatacaaaaattagccg
ggcgtggtggcgcgccctgtaatcccagcactcgggaggctgaggcaggagaatcgcttgaacccggg
aggcggaggttgcagtgagccgagatcgcgccactgcactccagcctgggacagagcgagactccgctc
tcaaaaaaaaa
```

Most of PFP's internal defaults are unmodified and the program is executed with a very basic command-line specification. The command-line to run is:

```
scylla -Query=test.fasta -Reference=alu.fasta -Output=masked.out -Stat=stat.out
-Debris=debris.out
```

This command specifies that we are running a single search stage of `test.fasta` against `alu.fasta`. The output file, `masked.out`, will contain the annotated sequences. The action to take on these sequences is, by internal default, to mask the contaminant regions. Statistics are saved to `stat.out` and debris to `debris.out`.

Once this command has completed, then these three files--`masked.out`, `stat.out`, and `debris.out`--have been created on disk. The following sections are the contents of these files.

Example 2: CAML Annotation Using a Parameter File

This example will demonstrate a more complex usage of PFP. In this example, we want to annotate some human data that has been stored in CAML form. This data will be passed through multiple masking and filtering stages in order to fully clean it.

The parameters file that is similar to the example used in the previous example. The file name is `scylla.prm`. It is reprinted here to be complete:

```
-GapOpen      -6
-GapExtn     -2
-Match       1
-Mismatch    -3
-WordLen     8
-Alg        Haste
-Action     annot
-Annot      mask
-Complement Y

%DUST
-Reference  dust
-Threshold  25

%ALUS
-Reference  alus.fasta
-Threshold  30

%HUMREP
-Reference  humrep.ref
-Threshold  30

%HUMOLD
-Reference  humold.fasta
-Complement N
-Threshold  25
-Mismatch  -1
-WordLen   6
```

PFP can then be run with the command:

```
scylla -Query=K03021.caml -Output=annot.caml -Param=scylla.prm
```

The query and output files are too large to print here in their entirety, but the following sections demonstrate how PFP annotates CAML files.

The input file, `K03021.caml`, is as follows:

```
<CAML>
<SEQUENCE NAME="HUMTPA">
  <BASE>
    ttcacacaactggtgctgttaccaccatgggctctagtctggatcagtggtcctcagtc
    ttttttgaccaggaccagttttgtaaagatagcttttccacggacagagggaggggag
    ...
    aaggaaaaataactgggtgagacgtggactgtggacaggcgcggaagggcac
  </BASE>
</SEQUENCE>
</CAML>
```

As it passes through PFP, this data is incrementally annotated and eventually written to the output file named `annot.cam1`. This output file is shown below:

```
<CAML>
<SEQUENCE NAME="HUMTPA">
  <BASE>
    ttcacacaactgggtgctgttaccaccatgggcgtctagtctggatcagtggtcctcagtc
    ttttttgcaccagggaccagttttgtaaagatagcttttccacggacagagggaggggag
    ...
    aaggaaaaataactgggtgagacgtggactgtggacagcgcggaaggcac
  </BASE>
  <EDIT QSTART="1014" QEND="1046" SCORE="93" SOURCE="scylla" DSTART="0"
    DINDEX="0" ACTION="mask" DEND="32" DORIENT="FORWARD"></EDIT>
  <EDIT QSTART="1407" QEND="1437" SCORE="34" SOURCE="scylla" DSTART="0"
    DINDEX="0" ACTION="mask" DEND="30" DORIENT="FORWARD"></EDIT>
  <EDIT QSTART="5668" QEND="5677" SCORE="31" SOURCE="scylla" DSTART="0"
    DINDEX="0" ACTION="mask" DEND="9" DORIENT="FORWARD"></EDIT>
  <EDIT QSTART="6465" QEND="6489" SCORE="57" SOURCE="scylla" DSTART="0"
    DINDEX="0" ACTION="mask" DEND="24" DORIENT="FORWARD"></EDIT>
  ...
  <EDIT QSTART="22758" QEND="22872" SCORE="55" SOURCE="scylla" DSTART="115"
    DINDEX="0" ACTION="mask" DEND="232" DORIENT="FORWARD"></EDIT>
</SEQUENCE>
</CAML>
```

Note the number of `EDIT` tags that appear within the `SEQUENCE` tags. Each of these corresponds to a single region that PFP found during a stage. Each `EDIT` specifies the bounds within the query for the annotation, a score, information on the reference document generating the hit, a source that generated the `EDIT`, and an action to take. The annotations have been given the name `EDIT` to comply with a more general representation of actions that can be performed on a sequence. The `ACTION="mask"` tag specifies that the user set the action to perform. From these bits of information, downstream applications can choose whether or not to actually apply the `EDIT`.

The statistics for this larger run are also important. Here is an abridged listing of `stat.out`:

QName	QStart	->	QEnd	DName	DStart	->	DEnd	Score	
HUMTPA+	1014	->	1046	dust+	0	->	32	93	
HUMTPA+	1407	->	1437	dust+	0	->	30	34	
HUMTPA+	5668	->	5677	dust+	0	->	9	31	
HUMTPA+	6465	->	6489	dust+	0	->	24	57	
HUMTPA+	7163	->	7224	dust+	0	->	61	60	
...									
HUMTPA+	742	->	1013	AluSg+	4	->	279	144	
HUMTPA+	5678	->	5947	AluSq-	2	->	272	164	
HUMTPA+	6318	->	6462	AluSg-	133	->	281	85	
HUMTPA+	6495	->	6641	AluYb5-	13	->	160	45	
HUMTPA+	6512	->	6641	AluSq-	27	->	156	78	
HUMTPA+	6639	->	6735	AluSz-	177	->	273	69	
HUMTPA+	7252	->	7354	AluJo-	24	->	127	37	
HUMTPA+	7387	->	7500	AluJo-	159	->	272	38	
...									
HUMTPA+	45	->	534	MER1A MER1A	No-	0	->	489	218
HUMTPA+	109	->	534	MER1A MER1A	No-	64	->	489	242

```

HUMTPA+      1846 -> 1919      L1PA11|L1PA11  -      387 -> 460      30
HUMTPA+      5305 -> 5557      MER4A|MER4A  in-      0 -> 254      129
HUMTPA+      5960 -> 6295      MER4A|MER4A  in-      322 -> 657      223
...

```

Stage 0: dust

Total number of Query bases input to Paracel Filtering Package: 36595

Reference Seq Name	#Hits	Tot Len	Query%
dust	29	1355	3.703%

Total	29	1355	3.703%

Stage 1: alus.fasta

Total number of Query bases input to Paracel Filtering Package: 36595

Reference Seq Name	#Hits	Tot Len	Query%
FLA	1	81	0.221%
AluJo	14	1628	4.449%
AluJb	6	807	2.205%
AluSz	5	965	2.637%
AluSx	2	408	1.115%
AluSg	2	417	1.139%
AluSq	6	1091	2.981%
AluSp	3	599	1.637%
AluSc	1	68	0.186%
AluY	1	270	0.738%
AluYa1	0	0	0.000%
AluYb1	0	0	0.000%
AluYa5	0	0	0.000%
AluYb5	1	147	0.402%
AluYa8	1	280	0.765%
AluYb8	1	235	0.642%
LTR26B	0	0	0.000%
LTR26C	0	0	0.000%
LTR26D	0	0	0.000%

Total	44	6996	19.117%

Stage 2: humrep.ref

Total number of Query bases input to Paracel Filtering Package: 36595

Reference Seq Name	#Hits	Tot Len	Query%
ALU ALU Human ALU intersperse	0	0	0.000%
MIR MIR Mammalian-wide inters	0	0	0.000%
L1PA2 L1PA2 3'-end of L1 repe	0	0	0.000%
L1PA7 L1PA7 3'-end of L1 repe	0	0	0.000%
L1PA11 L1PA11 3'-end of L1 re	1	74	0.202%
...			
MER1A MER1A Nonautonomous DNA	2	916	2.503%
MER1B MER1B Nonautonomous DNA	0	0	0.000%
MER2 MER2 Nonautonomous DNA t	1	176	0.481%
MER3 MER3 Nonautonomous DNA t	0	0	0.000%

```

MER4A|MER4A internal sequence      4      1204      3.290%
...
-----
                        Total      12      2833      7.741%
-----

-----
Stage 3: humold.fasta
-----
Total number of Query bases input to Paracel Filtering Package: 36595

Reference Seq Name      #Hits      Tot Len      Query%
      MIR@1              1          115          0.314%
      LINE2@1            0           0          0.000%
      MIR@2              0           0          0.000%
      LINE2@2            0           0          0.000%
-----
                        Total      1          115          0.314%
-----

```

From this statistics file we can see that PFP went through four stages of cleaning of this input file. At the beginning of the file there is a list of all the hits, in the order of the stage and then in the order of their position within the query. The hits start with dust hits, then move on to Alus and other human repetitive elements. The stage statistics indicate how much of the query was masked at each stage. This is useful information to get a rough idea of PFP's performance. In evaluating the results, the user can look at the total number of bases masked for a given stage and compare that with his or her expectations for that type of contaminant sequence.

File Management

This appendix describes utilities that are included with Paracel Filtering Package, but are not usually necessary during normal execution. You may find these scripts and executables helpful to set up PFP or perform analysis on the results. These utilities can be found in the `script/` directory or in one of the `bin.*/` directories in the System Installation.

The sequtil Utility

The `sequtil` utility included with PFP has a variety of purposes: concatenating sequence files of varied formats, retrieving statistics (e.g. number of sequences, total number of bases, average sequence length) from sequence files, sampling sequences out of files, and selecting sequences out of files.

Usage

```
sequtil [files...] [options...]
```

Description

The `sequtil` utility takes any number of sequence files or directories containing sequence files as input and performs a variety of tasks on those files. The files must all

be listed before options and can be in any of the supported formats: CAML, FASTA, GenBank, EMBL, phd, IG, GCG. If a directory is specified, `sequtil` will use all files in that directory.

This utility has a variety of purposes: concatenating sequence files of varied formats, retrieving statistics (e.g. number of sequences, total number of bases, average sequence length) from sequence files, sampling sequences out of files, and selecting sequences out of files.

Arguments

- edit
(Optional) For CAML input files, this parameter applies sequence EDIT tags to the sequence data when reading the file.
- fmtstr
(Optional) This argument sets the fields to display in the output when the `-stat` flag is set. It expects a case-insensitive string consisting of any of the following characters: F (filename), T (file type), B (#bases), S (#seq), A (avg len), I (min len), X (max len), D (median len), G (GC content), N (N content). Default is FTBSAIX.
- format <file_format>
(Optional) This parameter sets the format of the output file. Valid values are: `fasta`, `caml`, `ig`, `gb` (GenBank), `gcg`, `embl`, `phd`, `bin` (binary), and `cbin` (compressed Binary). Note that if the file type is FASTA, `sequtil` will look for a corresponding quality value file. These files must be named according to the following convention: `<filename>.fasta` for the sequence data and `<filename>.fasta.qual` for the quality values.
- invert
(Optional) Inverts the selection made using `-select`, meaning that sequences that are not in the `-select` list will be used.
- out <filename>
(Optional) Specifies the output file name. `sequtil` will write the processed input sequences to this output file. If this option is not specified, then `sequtil` turns the `-stat` option on by default. The default format of the output file is the format of the first input file. This can be overridden by using the `-format` option.
- qual
(Optional) When the output file format is FASTA, this parameter specifies that `sequtil` write out a `.qual` file in addition to the FASTA file. This `.qual` file will hold the quality values of the sequences in FASTA-Qual format. A default QV of 10 is used if a sequence does not have quality values.
- revcomp
Specifies to output the reverse complement of all of the sequences written to the output file. Note that this parameter only works on nucleotide sequences.

- `-sample <percent>` (Optional) Samples sequences from each input file. Each sequence in the input file has the specified percent chance of being selected. For example, `-sample 5` would specify that each sequence has a 5% chance of being selected.
- `-select <list>` (Optional) Selects a list of sequences from the input file. The list must be comma-separated and can support ranges (e.g. 1, 2, 5-10, 50-66). The indices used to select are indexed from zero in each input sequence file. This means that if `-select 2-5` is specified and multiple input sequences are specified as well, then sequences 2-5 from each input file will be used. Thus, applying this option to multiple input files may not be desirable. Sequence indices can be placed into a separate file and the file can be given to the `-select` option if prefixed by '@', e.g., `-select 1,2,5-10,@seqnums`. Sequence indices in the file must be separated by whitespace.
- `-stat`
- `-winlen <window length>` (Optional) Breaks the input sequences into windows of specified size. Size is specified in number of bases.
- `-winovr <overlap length>` (Optional) Used in conjunction with the `-winlen` parameter, it specifies the overlap, in number of bases, between adjacent windows.

Examples

1. `sequtil rat.fasta mouse.gb cow.embl`

This example specifies to return the statistical data for the three specified input files (in different formats).

2. `sequtil rat.fasta -out rat.gb -format gb`

This example specifies to convert the file `rat.fasta` into a GenBank format file named `rat.gb`.

3. `sequtil rat*.fasta -out allrat.fasta`

This example concatenates all of the files with names matching `rat*.fasta` into a file named `allrat.fasta`.

4. `sequtil allhuman.fasta -sample 10 -out human10.fasta`

This example samples approximately 10% of the sequences from the file named `allhuman.fasta` and writes them to a file named `human10.fasta`.

5. `echo "10 13 65 99" > seqs`

```
sequtil rat.fasta -select 0,5-9,@seqs -out select.fasta
```

This example selects the sequences with the following index numbers from the file `rat.fasta`: 0, 5, 6, 7, 8, 9, 10, 13, 65, 99. These sequences are then written to the file named `select.fasta`.

```
6. sequtil rat.fasta -select 0,1 -invert -out select.fasta
```

This example writes every sequence, except for the first two sequences, from the file `rat.fasta` to the file `select.fasta`.

```
7. sequtil rat.phd -out rat.fasta -qual -format fasta
```

This example writes all sequences in the file `rat.phd` to the file `rat.fasta` and their associated quality values to the file `rat.fasta.qual`.

rebaseupdate

Usage

```
rebaseupdate rebasedir installdir
```

Description

This utility updates the System Installation of PFP at the specified *installdir* with a new copy of the RepBase reference files located in the specified *rebasedir*. The reference files should have `.ref` file extensions, just as those on the RepBase CD-ROM distributed by Paracel. This script modifies the System Installation so that PFP will use the new RepBase files instead of the public reference files.

Arguments

<i>rebasedir</i>	Directory in which the RepBase files are stored. This directory should contain the <code>.ref</code> files contained in the RepBase distribution. For example, <code>athrep.ref</code> , <code>humrep.ref</code> . The <i>rebasedir</i> is a directory on the Rebase CD ending in <code>.fasta</code> , such as <code>/cdrom/cdrom#/rebase.x.fasta</code> , where <i>x</i> is the version number.
<i>installdir</i>	Directory where the installation of PFP resides. The <i>installdir</i> is the location of the System Installation, i.e. the setting of <code>PTAHOME</code> .

Important Note: This script looks for the following files under *rebasedir*: `README.html`, `README.txt`, `athrep.ref`, `celrep.ref`, `drorep.ref`,

humrep.ref, humsub.ref, invrep.ref, mamrep.ref, plnrep.ref, psuedo.ref, rodrep.ref, simple.ref, vrtrep.ref and zebrrrep.ref.

These files are copied into the `installdir/reference/rebase` directory. The script then concatenates all reference files for a particular organism to a file name recognized by TranscriptAssembler. The files created under the `installdir` are:

TABLE A-1: Rebase Reference Files

<i>Reference File</i>	<i>Concatenated Rebase Files</i>
reference/human.repeats	humrep.ref and humsub.ref
reference/rat.repeats	rodrep.ref
reference/mouse.repeats	rodrep.ref
reference/danr.repeats	zebrep.ref
reference/aratha.repeats	athrep.ref
reference/celeg.repeats	celrep.ref
reference/dros.repeats	drorep.ref

Examples

```
setenv PFPHOME /paracel/paracel/pfp/CurrentRelease
rebaseupdate /cdrom/cdrom#1/rebase.x.fasta $PFPHOME
```

The first line in this example sets the `PTAHOME` environment variable to the location of the PFP System Installation. The second line calls the `rebaseupdate` script with the location of the CD-ROM containing the `.ref` RepBase files. `rebaseupdate` uses the setting of `PTAHOME` to find the PFP installation to update.

Index

A

Actions
 Annotate 1-4
 Excise 1-4
 Filter 1-4
 Mask 1-3
Annotate 1-4
Atail 3-3
ATAIL algorithm 5-1

C

CAML 1-2
Cloning vector contaminants 5-5
Customer Support 1-6

D

DUST 1-2, 3-3
DUST algorithm 1-3, 5-2

E

E. coli 1-2, 5-5
EMBL 1-2

F

FASTA 1-2
File formats 1-2

G

Gap penalties
 Extension 3-4, 5-5
 Open 3-4, 5-5
GenBank 1-2

H

HASTE search algorithm 5-6

I

IG 1-2
Input file formats 1-2

L

LINES 1-2
Long Interspersed Elements 1-2
Low complexity regions 1-2
Low quality regions 1-3
Low-complexity annotation 5-2

M

Masking characters 1-3, 3-5
Match score 3-5, 5-5
Maxlen 3-3
Minimum sequence length 1-3
Minlen 3-3
Mismatch score 3-5, 5-5
Mitochondrial data 5-5

O

Organism-specific parameter files 6-1

P

Parameter files 6-1
 Editing 6-2
 Examples 6-5
 Organism-specific 6-1
 Specifying 3-5
Parameter parsing rules 6-4

- PFP parameters
 - Action 1-3
 - Annot 1-4
 - Debris 1-4
 - Stat 1-5
- Pfpsetup.pl 2-2
- PFPView 1-1, 4-1
- Phd 1-2
- Platforms 2-1
- Poly-A 1-3
- Poly-T 1-3
- Profile searches 5-6
- PSEG 3-3

- Q**
- Qualclean 3-3
- Quality-value cleaning method 5-4
- Query 1-2

- R**
- Reference file
 - aratha.repeats A-5
 - celeg.repeats A-5
 - danr.repeats A-5
 - dros.repeats A-5
 - human.repeats A-5
 - mouse.repeats A-5
 - rat.repeats A-5
- RepBase 1-3, 5-5
- rebaseupdate A-4
- Repeats
 - poly-A 5-1
 - poly-T 5-1
- Repetitive/contaminant region algorithms
 - ATAIL 5-1
 - DUST 5-2
 - MAXLEN 5-3
 - MINLEN 5-2
 - PSEG 5-3
 - QUALCLEAN 5-4
 - SEARCH 5-5
 - TRIMJUNK 5-4

- S**
- Score threshold 5-5

- Scoring matrix 5-5
- Scylla 1-1, 3-2
- Scylla parameters
 - Action 3-3, 5-2
 - Alg 3-3, 5-5
 - align 3-3, 5-7
 - AlignOut 3-4
 - Annot 3-4
 - Complement 3-4, 5-5
 - DBHost 3-4, 5-7
 - Debris 3-4
 - GapExtn 3-4, 5-5
 - GapOpen 3-4, 5-5
 - GMDir 3-4
 - GMMountPoint 3-4, 5-7
 - JunkChars 3-5, 5-2, 5-3, 5-4
 - MaskChar 3-5
 - Match 3-5, 5-1, 5-5
 - Matrix 3-5, 5-1, 5-5
 - Mismatch 3-5, 5-1, 5-5
 - NoEdits 3-5
 - NumProc 3-5
 - Output 3-5
 - Param 3-5
 - PFPView 3-6
 - PolyDist 3-6, 5-1
 - Priority 3-6, 5-7
 - PSegHighCut 5-3
 - PSegLenMin 5-3
 - PSegLowCut 5-3
 - PSegWin 5-3
 - Query 3-6
 - Reference 3-6, 5-5
 - Stat 3-6
 - TempDir 3-6
 - Threshold 3-6, 5-1, 5-5
 - Verbosity 3-7
 - WordLen 3-7, 5-7
 - WProfThresh 3-7, 5-6
- Search algorithms
 - GMPROF 5-6
 - HASTE 5-6, 5-7
 - Smith-Waterman 5-8
- SEG 1-2
- Sequitil 1-1

Short Interspersed Elements 1-2

SINES 1-2

Stages of execution 1-3, 6-3

Statistics file 1-5

System installation script 2-2

T

Trimjunk 3-3

U

UniVec 1-2

Utility program `sequtil` A-1

V

Vector 1-2

