CONSED 15.0 DOCUMENTATION

CONTENTS:

WHAT IS NEW IN CONSED 15.0

AMD 64 Bit Linux Now Available

Linux version more stable
There are actually many different versions of linux, and incompatibilities between consed and some of them caused problems. I believe all of these problems are solved.

For Rapid Review of Many Different Projects
Can write a script that starts up consed repeatedly on different projects, each time with a custom navigation file loaded

```
consed -ace (ace file) -nav (custom navigation file)
```

Assembly View
Shows Restriction Fragments (and shows which ones don't match the gel).

Can find a contig by typing its name--useful for projects with hundreds or thousands of contigs

Traces Window
Shows Consensus Tags

Autofinish
improvements in PCR

Autofinish and Assembly View
Can import PCAP contig ordering information

List of Contigs in Main Window
Can be reordered by # of reads

AutoReport feature
Print out assembly information, such as the list of scaffolds (how the contigs are ordered and oriented)

Show All Traces vertical scrolling
E.g., can move by exactly one screenful of traces

Many bugs fixed

INSTALLING CONSED

You MUST have the following phred, phrap, phd2fasta, and crossmatch in order to use this version of Consed:

000925.c or later for phred
0.990319 or later for phrap and crossmatch
0.990622.e or later for phd2fasta (supplied with this version of consed)
any version of addReads2Consed.perl (supplied with this version of consed)
030415 of phredPhrap (supplied with this version of consed)
(Note: if you have an older version of phredPhrap, some of the more recent Consed features, such as miniassemblies, will not work. Note to existing polyphred users: phredPhrap now calls polyphred with different parameters which will give cause it to apply different tags than it used to, but these different tags will give it behavior consistent with that described below in CONSED-POLYPHRED INTERACTION. For more information, see http://droog.gs.washington.edu/PolyPhred.html)

030117 or later for transferConsensusTags.perl (supplied with this version of consed)
any version of tagRepeats.perl (supplied with this version of consed)
any version of determineReadTypes.perl (or your own custom modified version)

For phred, contact bge@u.washington.edu (Brent Ewing)

For phrap and crossmatch, contact phg@u.washington.edu (Phil Green)

Summary of files your must edit (instructions are below):

```
          addReads2Consed.perl
          determineReadTypes.perl
          phredPhrap
          primerCloneScreen.seq
          primerSubcloneScreen.seq
          repeats.fasta
          vector.seq
```

In order to run the gauntlet of phred/phd2fasta/crossmatch/phrap, there is a perl script phredPhrap supplied with Consed (above). YOU MUST USE THIS PERL SCRIPT. If you try to run each of these programs directly, you are on your own and you will probably spend a lot of time needlessly.

1) After downloading the distribution with netscape (see www.phrap.org and click on 'Consed'), copy the distribution to a unix computer (if it is not already on one). Unpack the files by typing the appropriate line below (which one depends on what you named the file downloaded by netscape):

```
zcat consed_linux.tar.Z    | tar -xvf -
zcat consed_linux_itanium.tar.Z    | tar -xvf -
zcat consed_solaris.tar.Z | tar -xvf -
zcat consed_macosx.tar.Z  | tar -xvf -
zcat consed_amd64.tar.Z    | tar -xvf -
zcat consed_alpha.tar.Z    | tar -xvf -
zcat consed_sgi.tar.Z      | tar -xvf -
zcat consed_ibm.tar.Z      | tar -xvf -
zcat consed_solaris_intel.tar.Z | tar -xvf -
zcat consed_hp.tar.Z       | tar -xvf -
```

Note: You must run tar on a UNIX computer--not on an Windows computer, due to a difference in the handling of breaks between lines.

2) I suggest you put Consed, phred, crossmatch, phrap, the perl scripts, and other executables into /usr/local/genome/bin. So create /usr/local/genome/bin and /usr/local/genome/lib

If you can't actually use /usr/local/genome, then you could make /usr/local/genome be a link to the real location--that will work just as well.

If you want to have another location xxx, then put:

```
setenv CONSED_HOME xxx
```

into the .cshrc (or equivalent if you are using bash or a shell other than csh or tcsh) of all Consed users

and create $CONSED_HOME/bin and $CONSED_HOME/lib and put all of these programs into $CONSED_HOME/bin

3) Make sure that /usr/local/genome/bin (or $CONSED_HOME/bin) is in every Consed users' PATH.

4) Put the Consed executable in /usr/local/genome/bin (or $CONSED_HOME/bin)

Read the appropriate section of this document: NOTE TO SOLARIS USERS, NOTE TO SGI USERS, NOTE TO MACOSX USERS, NOTE TO LINUX USERS, or (if you are running Linux on an Itanium--a big 64 bit box) NOTE TO ITANIUM LINUX USERS.

5) Check this by logging on as a user and typing:

```
(consed executable name) -V
```

where (consed_executalbe name) is one of:

```
consed_linux2.4
consed_linux2.6
consed_linux2.6_dyn
consed_solaris
consed_mac
consed_linux_itanium
consed_hp
consed_solaris_64
consed_amd64
consed_ibm
consed_alpha
consed_solaris_intel
consed_sgi
```

You should see 'Version 15.0'. If you see something else, you have some debugging to do.

6) TESTING CONSED

Follow the first few steps of USING CONSED GRAPHICALLY of the QUICK TOUR (below). If you have problems, it may be due to your X emulator. See 'MONITORS AND MICE FOR CONSED' below.

If you get some error such as:

```
Error: Can't open display:
```

then the problem may have nothing to do with Consed, but rather with X. To test this, run some other X application (such as xclock, xterm, xeyes, or xcalc) and see if you get the same error.

7) Build phd2fasta:
Go to the misc/phd2fasta directory and type 'make'
Move the phd2fasta executable to /usr/local/genome/bin (or $CONSED_HOME/bin)

8) Build mktrace:
Go to the misc/mktrace directory and type 'make'
Move the mktrace executable to /usr/local/genome/bin (or $CONSED_HOME/bin)

9) Move all perl scripts from the scripts directory to
/usr/local/genome/bin (or $CONSED_HOME/bin)
Make sure all are executable (chmod a+x *)

10) Get perl 5. (If you have Linux, you already have it so you can
skip this step.) You can check where to get perl via the perl web
site:

```
     http://www.perl.com
```

(If you don't know about perl, try it--it will save you a
huge amount of time over developing the same utilities in C, awk, or
csh or sh.)

Regardless where you put perl, put a link to it in /usr/bin so that
all of the scripts with
#!/usr/bin/perl
will work and you won't have to edit all of them everytime a new
Consed release comes out.

11) Create a subdirectory /usr/local/genome/lib/screenLibs. (If you
are using a location other than /usr/local/genome for the root of all
Phred/Phrap/Consed programs, create $CONSED_HOME/lib/screenLibs). From
the misc subdirectory, copy primerCloneScreen.seq and
primerSubcloneScreen.seq to the directory
/usr/local/genome/lib/screenLibs (or $CONSED_HOME/lib/screenLibs).

primerCloneScreen.seq is used to screen candidate primers when you use
Consed's function "Pick Primer from Clone Template" (on the Aligned
Reads Window).

primerSubcloneScreen.seq is used to screen candidate primers when you
use Consed's function "Pick Primer from Subclone Template" (on the
Aligned Reads Window).

Take a look at these files. They are dummy files indicating the fasta
format of the sequences that should be put in them. You should put
into primerCloneScreen.seq the vector sequence of the cloning vectors
you are using (BAC or cosmid) and into primerSubcloneScreen.seq the
sequencing vectors you are using (plasmid, M13, etc). Don't be too
generous in putting lots of vectors into the files! The larger they
are, the slower primer picking will be. Our files are only this big:

```
-rw-r--r--   1 root     root        29938 Nov  7  1997 primerCloneScreen.seq
-rw-r--r--   1 root     root         7381 Aug 13  1997 primerSubcloneScreen.seq
```

and primer picking is quite fast enough.

Now that you have set this up, you should try the PRIMER PICKING
sections in the Quick Tour (above) to make sure this works.

12) You should create a file

/usr/local/genome/lib/screenLibs/vector.seq

(or $CONSED_HOME/lib/screenLibs/vector.seq if you are not using
/usr/local/genome for the root of the Phred/Phrap/Consed files.)

This contains all the vector sequences (in FASTA format) that you want
to mask out before running phrap. In general, it is the combination of
primerCloneScreen.seq and primerSubcloneScreen.seq.

13) You should create a file
/usr/local/genome/lib/screenLibs/repeats.fasta

(or $CONSED_HOME/lib/screenLibs/repeats.fasta if you are not using
/usr/local/genome for the root of the Phred/Phrap/Consed files.)

In this file, put any sequences (in FASTA format) that you want to
have automatically tagged. These typically are ALU sequences. If you
don't want to tag anything, then comment out (put '#' as the first
character of the line) the following lines in phredPhrap:

Change:
```
!system( "$tagRepeats $szAceFileToBeProduced" )
   || die "some problem running $tagRepeats";
```

to:
```
#!system( "$tagRepeats $szAceFileToBeProduced" )
#   || die "some problem running $tagRepeats";
```

14) You should create a file
/usr/local/genome/lib/screenLibs/singleVectorForRestrictionDigest.fasta
containing the cloning vector sequence. This is used for doing in-silico
restriction digests. Thus this cloning vector must start at precisely
the site where you cut the vector to ligate the insert. It is not
sufficient to just download the vector sequence from Genbank. You
might need to have it start in a different place.

15) SETTING UP TEST DIRECTORIES

Copy the test directories and their contents to some location where
the users have write access. Copy--do not move them--because the
users will occasionally want a fresh copy.

```
cp -r standard new_location
cp -r autofinish new_location
cp -r assembly_view new_location
cp -r polyphred new_location

cd new_location
chmod -R a+w *
```

16) MODIFYING determineReadTypes.perl

Read the comments in determineReadTypes.perl

Phrap, Consed's primer picking, and Consed/Autofinish all need the following information for each read:

```
             is it a univeral primer forward, a universal primer reverse,
                or a walking read?
            what is its template name?
```

If you are using different libraries that have different insert sizes, then Consed/Autofinish also need the library name for each read.

Generally this information can be determined from the read name, using *your* naming convention. Modify the perl script determineReadTypes.perl to put this information at the end of the phd file using WR info items.

If you don't want to do much perl programming and all your libraries have the same insert size, you have the option of using the St Louis naming convention. In this case, the only perl programming you need do is to comment out (put a "#" in front) the line in determineReadTypes.perl that starts with:

```
die "You must edit determineReadTypes.perl
```

You must also uncomment (remove the "#"s in column 1) the lines in the phredPhrap script that say roughly:

```
#print  "\n\n---------------------------------------------------------\n";
#print  "Now running determineReadTypes.perl...\n";
#print  "---------------------------------------------------------\n\n\n";

#!system( "$determineReadTypes" ) || die "some problem running determineReadTypes.perl $!\n";
```

But what is the St Louis naming convention? Most of it (but not all) is explaned in the phrap documentation. In addition, you must never use an underscore in the name if the read is a universal primer forward or universal primer reverse read. If the read is a walk, then you must have an underscore (_) follow the template name and then have a number (the oligo number).

Examples of reads in the St Louis naming convention:

```
read eeq03a01.g1.phd.1 is univ rev template: eeq03a01 library: eeq03
read eeq03a02.b1.phd.1 is univ fwd template: eeq03a02 library: eeq03
read eeq03a02.g1.phd.1 is univ rev template: eeq03a02 library: eeq03
read eeq03a03.b1.phd.1 is univ fwd template: eeq03a03 library: eeq03
read eej45h07_2.i1.phd.1 is walk template: eej45h07 library: eej45
read eej46c12_1.i1.phd.1 is walk template: eej46c12 library: eej46
```

Once you have correctly customized determineReadTypes.perl, then uncomment the line in phredPhrap which calls determineReadTypes.perl

It is fine to assume the St Louis naming convention for the purpose of the sample dataset directories that come with Consed ("standard", "assembly_view", "autofinish", and "polyphred").

17) TROUBLESHOOTING YOUR CHANGES TO determineReadTypes.perl

Consed allows you to check that you have correctly modified determineReadTypes.perl: On the Consed Main Window, point to 'Info', hold down the left mouse button, and release on 'Show Info for Each Read'. Study all the information and check that the information presented is correct. If, for example, Consed thinks that there are templates that have 9 or more reads, it is likely that you have not correctly customized determineReadTypes.perl

You will see a section that looks like this:

```
template djs736a2_fp04q286 with 2 reads
       djs736a2_fp04q286.x2 term      universal forward (from phd file)
       djs736a2_fp04q286.y2 term      universal reverse (from phd file)
```

You want to see the "from phd file" part. If, instead of "from phd file", it says "inferred from name", that means that determineReadTypes.perl couldn't figure out what kind of read it was.

If you think you have made a mistake in customizing determineReadTypes.perl, it is best to delete the PHD files (and phd.ball if you are using that) and run phredPhrap again since the otherwise incorrect WR items will be left in the PHD files.

There is more specific documentation within the script determineReadTypes.perl for more information about how to customize it.

CUSTOMIZING determineReadTypes.perl: SPECIAL CASES

18) FAKE READS

By "fake reads" I mean reads such as those created from a Genbank reference sequence or a consensus from some other assembly... or others for which there is no chromatogram (and there never was any chromatogram). If you don't use any such reads, you can skip this step.

In the past, any read that ended with a .a2 or .c3 (where 2 and 3 could be any numbers), was considered a fake read. Now you can make Autofinish not assume this using the .consedrc parameter (see CONSED CUSTOMIZATION):

```
consed.fakeReadsSpecifiedByFilenameExtension: false
```

Instead, you must have determineReadTypes.perl put "fake" into the "type:" field of a "template" WR item. See determineReadTypes.perl for more information.

After installing Consed, you should run all the following tests to make sure you have installed everything correctly:

19) APPENDING EXPID TO THE PHD FILES

If you are using Autofinish, and would like Autofinish to tell you how well your reads are succeeding, then the phd files must be appended with the experiment id's. In the 3 Autofinish summary files (*.univReverse, *.univForwards, and *.customPrimers), you will see information like this:

```
univ  rev,,,->,-329,-249,71,Contig1,3,djs228_1034
```

or this:

```
tgaagaaatggctgactcc,56,1,->,3258,3338,3658,Contig1,4,djs228_2813,5,djs228_168,6,djs228_1248
```

The '3' just before the djs228_1034 on the line starting with "univ rev" is an experiment id. There is also an expid '4' just before djs228_2813, an expid '5' before djs228_168, and an expid '6' just before djs228_1248.

Autofinish doesn't know what you will end up calling these reads it is telling you to make. Autofinish only knows those reads by the numbers 3, 4, 5, and 6. So when you make the reads, Autofinish needs to be informed that this is 'experiment 3' or whatever. You do this by appending in the phd file the following structure:

```
WR{
expid addExpid 990811:140818
5
}
```

```
where WR stands for 'whole read item',
       expid for 'expid'
        addExpid is the name of the program that you will write that
               will append this information
        990811:140818 is the date and time in format YYMMDD:HHMISS
       5 is the expid
```

This program must be run *after* phred runs to create the phd files. Thus your program must have some method of determining what the expid of each read is. What the University of Washington Genome Center does is to have the finishers put the expid as part of the filename. This makes it easy for a program to look at the phd file and figure out what the expid is and then write the WR item into that phd file.

Alternatively, you could keep a database and, after the phd file is created, look into the database to see what the expid is.

When you have successfully added expid's to the phd files, the next time you run Autofinish on this project, it will have in the 'EVALUATE' section of the Autofinish output file, lots of interesting information about how well the reads succeeded.

TESTING

The following tests must be done to insure that the installation was done correctly.

20)   TESTING RESTRICTION DIGEST

Try the restriction digest feature (RESTRICTION DIGEST above) to make sure this works.

21)   TESTING ADD NEW READS

It will make your life easier if phred, phrap, and crossmatch are all where Consed expects them: in /usr/local/genome/bin

22) Decide where to put phred's parameter file phredpar.dat and edit both addReads2Consed.perl and phredPhrap to reflect this location. I generally prefer to put it in /usr/local/genome/lib to keep all of the Phred/Phrap/Consed files in one place. Alternatively, you could put it in /usr/local/etc/PhredPar/phredpar.dat which is the historical location of this file.

23) Next you should test the ADD NEW READS step in the Quick Tour (below). This step requires that everything be set up correctly and in the correct location. Hopefully the error messages are clear enough to help you if you have set up anything incorrectly.

24)   TESTING RUNNING CROSSMATCH FROM ASSEMBLY VIEW

See RUNNING CROSSMATCH FOR SEQUENCE MATCHES (above) and make sure that step works.

25)   TEST RUNNING PHREDPHRAP

See the section RUNNING PHRED and PHRAP in the Quick Tour (below)

26)   TESTING MINIASSEMBLIES

See PULLING OUT READS AND RE-ASSEMBLYING THEM (MINIASSEMBLIES) and MINIASSEMBLIES (below) and make sure those steps work.

The newer version of phredPhrap is required for this. If you have invested a lot of work customizing some old version of phredPhrap, and don't want to upgrade, you do have the option of keeping your customized version of phredPhrap for regular assemblies, and using the new version of phredPhrap for miniassemblies. To do this, you must specify the alternate name/location of phredPhrap by the .consedrc parameter:

```
consed.fullPathnameOfMiniassemblyScript:  /usr/local/genome/bin/phredPhrap
```

(See CONSED CUSTOMIZATION below.)

USING YOUR OWN DATA

27) Create the following directory structure, which can be anywhere on any disk:

```
Directory structure:
     top level directory (generally named after the BAC or cosmid)
          subdirectory 'chromat_dir'--chromatograms go in here
          subdirectory 'phd_dir'--phd files will automatically be put here
          subdirectory 'edit_dir'--ace files will automatically be put here
```

Put the chromatogram files (e.g., .ab1 or .scf files) into the chromat_dir directory. Keep phd_dir and edit_dir empty.

If you already have your chromatograms somewhere else, you can make chromat_dir be a link to wherever you have them.

The various phrap and crossmatch files will be put into edit_dir by the phredPhrap script.

28) cd to the edit_dir directory, and type:

```
phredPhrap
```

If you are successful, the script will tell you so. (You can also look in phd_dir and you will see phd files for each of the chromatograms you added in chromat_dir. If the phd files are missing, then phred was unable to call bases from the chromatograms in chromat_dir and you will need to figure out why not). Make sure you are in edit_dir and bring up Consed on the ace file:

29) Bring up Consed as shown in the "QUICK TOUR OF CONSED"

```
consed
```

You should see a file with the extension .ace.1
Double click on it.

You should see a list of contigs.

Double click on the one you want to see.

Follow the first few steps of the QUICK TOUR OF CONSED (below). You should at least go as far as viewing traces.

QUICK TOUR OF CONSED

Release 15.0

Consed is a program for viewing and editing assemblies assembled with the phrap assembly program.

If you are already an advanced Consed user, you should read through this and do any of the exercises on features that you are unfamiliar with. I frequently run across people who are doing something in Consed a hard way month after month, and request a new feature to make things easier, when that new feature is already in Consed.

If you have never used Consed before, to follow this Quick Tour will take you less than 6 hours. However, it will save you approximately 2 days in agony. If you have 2 extra days to spare, and prefer to waste them in agony, then do not do this Quick Tour and instead immediately skip down to 'INSTALLING CONSED' above.

If you do the Quick Tour, start your system administrator installing consed (see INSTALLING CONSED (above)) because you will need to have completed that for some of the more advanced sections of the Quick Tour.

When you do the quick tour, I encourage you to be free about changing the data set. If you really mess things up (such as changing all a read's bases to N's), no problem--just delete the data set and start again with a fresh copy.

USING CONSED GRAPHICALLY

30) Type the following:

```
cd standard/edit_dir
```

31) start Consed by typing the appropriate command below:

```
../../consed_solaris
../../consed_solaris64
../../consed_alpha
../../consed_hp
../../consed_sgi
../../consed_linux2.4
../../consed_linux2.6
../../consed_linux2.6_dyn
../../consed_mac
../../consed_linux_itanium
../../consed_solaris_intel
../../consed_ibm
../../consed_amd64
../../consed_amd64_dyn
```

(Don't worry about a message like:

```
Warning: Cannot convert string "helvetica" to type FontStruct )
```

Two windows will appear. One of these will have the list of .ace files and say 'select assembly file to open' and 'standard.fasta.screen.ace.1'. Double click on "standard.fasta.screen.ace.1". The first window goes away.

You will now see a list of one contig and a list of reads. This is the 'Consed Main Window'.

Double click on 'Contig1'.

The 'Aligned Reads Window' will appear.

32) SCROLLING

Try scrolling back and forth. Try scrolling by dragging the thumb of the scrollbar. Also try scrolling by clicking on the 4 buttons: << < > >> for scrolling by small amounts. For scrolling by tiny amounts, click on the arrows at either end of the scrollbar. For scrolling by huge amounts, use the middle mouse button and just click on some location on the scrollbar. For scrolling to the beginning or end of the contig, use the <<< or >>> buttons.

(Question: why can't you just move the scrollbar to the extreme right in order to go to the end of the contig? Answer: in typical assemblies, there are reads that protrude beyond the beginning of the contig and reads that protrude beyond the end of the contig. Moving the scrollbar to the extreme right will scroll the contig to the end of the rightmost read--typically far to the right of the end of the contig. Thus you should get in the habit of using the <<< and >>> buttons.)

GOTO POSITION

33) In the Aligned Reads Window, click in the 'Pos:' box in the upper right-hand corner. Type in a number, such as 540, and push the 'Return' or 'Enter' key. The Aligned Reads Window will scroll to position 540. We find this feature is particularly useful when one person wants another person to look at something in the sequence.

34) COLORS

Notice the colors. Scroll to position 937 and notice the read 'a'. The red bases are the ones that disagree with the consensus.

Notice the different shades of grey background (around the bases). They have the following meanings, but first, you need to understand the meaning of the quality values:

A quality value of 10 means 1 error in ten to the 1.0 power
A quality value of 20 means 1 error in ten to the 2.0 power
A quality value of 30 means 1 error in ten to the 3.0 power
A quality value of 40 means 1 error in ten to the 4.0 power

and for quality values in between:

A quality value of 25 means 1 error in ten to the 2.5 power

Get the idea?

(These have actually been empirically verified--if you are interested in the gory details, read the phred papers:

Ewing B, Hillier L, Wendl M, Green P: Basecalling of automated sequencer traces using phred. I. Accuracy assessment. Genome Research 8, 175-185 (1998).

Ewing B, Green P: Basecalling of automated sequencer traces using phred. II. Error probabilities. Genome Research 8, 186-194 (1998).

In that same copy of the journal is a paper about Consed, as well.)

Also notice the upper and lowercase. This is just a cruder indication of the quality of the bases.

35) To see the quality value of a particular base, point at it and click with the left mouse button. You will see the quality displayed in the Info Box at the bottom of the Aligned Reads Window.

These quality values are shown in grey scales:

Quality 0 through 4 is given by dark grey
Quality 5 through 9 is given by a shade lighter
Quality 10 through 14 is given by a shade still lighter
.
.
.
Quality of 40 through 97 is given by white (the brightest shade)

A quality value of 99 is reserved for bases that have been edited and the user is absolutely sure of the base ('high quality edited').

A quality value of 98 is reserved for bases that have been edited and the user is not sure of the base ('low quality edit').

The ends of the reads shows bases that are grey and have a black background. These are the low quality ends of the reads or the unaligned ends of reads, as determined by phrap.

36) Click on a base on a read. Then hold down the control key and type 'a'. You will move to the beginning of the read. Hold down the control key and type 'e'. You will move to the end of the read. (Emacs users will recognize these commands.)

37) HIGHLIGHTING READ NAMES

In the Aligned Reads Window, click on a read name with the left mouse button. The name will turn magenta. Click again and it will turn yellow again. Try turning it magenta and then scrolling. This feature is helpful in keeping track of a particular read as you scroll.

If you have an emacs window open (or any editor window), you can paste the read name in by just clicking with the middle mouse button. When you clicked on the read name in the Aligned Reads Window with the left mouse button, the read name was loaded into the paste buffer.

38) DIMMING ENDS OF READS

Scroll so that location 490 is about in the middle of the aligned reads window. Push the left mouse button down on the menu item 'Dim'. There will be a list of choices that will appear. Drag the cursor down to 'Dim Nothing' and release. Now look what happened to the color of the bases. The ends of the reads that used to be with a black background now appear red with a grey background. You are seeing the clipped-off bases with all the same information as any other base. Since there is a huge amount of red (discrepant) bases, the screen becomes distracting and busy. Thus by default the low quality clipped-off bases are made with a black background and a grey foreground so they don't distract you.

Notice there is a distinction here between 'low quality ends of reads' and 'unaligned ends of reads'. Unaligned ends of reads can be low quality as well, or they can be high quality, as in the case of chimeric reads.

Point with the mouse to a read name and hold down the right mouse button. You will notice there is a line that says "high quality from nnn to nnn; aligned from nnn to nnn; chem: prim". This is giving the same information in number form. Highlight the read name first (see HIGHLIGHTING READ NAMES above) so you don't lose the read as you scroll. Then check that the numbers agree with the dimming.

You can play with the dimming options a bit. Then return it to 'Dim Low Quality' for the rest of this tour.

TRACES AND EDITING

39) Point with the mouse at a base of one of the reads and click with the middle mouse button. (If you have a 2 button mouse, see MONITORS AND MICE FOR CONSED below.) The Trace Window showing the traces for that stretch of read should popup.

There are 2 rows of numbers:

```
'con' are the consensus positions
'rd'  are the read positions
```

There are 3 rows of bases in the trace window:

```
'con' is the consensus
'edt' is where you can edit the base calls of the read
'phd' is the original phred base calls
```

Notice that a red rectangle blinks (the 'cursor') in the corresponding positions of the Aligned Reads Window and the Trace Window.

40) Try editing in the Trace Window. You can click the left mouse button on a base in the 'edt' line to set the cursor (a blinking red rectangle). You can directly overstrike a base by typing a letter. Try this. Try undoing it (by clicking on 'undo'). If you want to undo more than one edit, you will have to go back to the main Consed window and click on the button labeled 'Undo Edit...'--you will learn that later. You can overstrike with the following characters: acgt (bases), * (a pad, in effect deleting the base), and mrwsykvhdb (IUB ambiguity codes).

You can move left and right with the arrow keys.

We believe that the user should change a base call only while examining the traces. That is why editing is done here--not in the Aligned Reads Window.

41) You can insert a column of pads by pushing the space bar. Try this. (You may need to click on a base on the 'edt' line first.)

(For those of you new to editing assemblies, a 'pad', which in Consed and phrap is represented by the '*' character, is used to align two or more sequences such as these:
```
      gttgacagtaatcta
      gttgacataatcta
```
in which one sequence has an inserted or deleted base with respect to the other. By inserting the pad character, it is possible to get a good alignment:
```
      gttgacagtaatcta
      gttgaca*taatcta
```
This is the purpose of pad character--it is just a placeholder.)

You can then overstrike a pad with a base. In this way you can insert a base, and still preserve the alignment.

42) Try highlighting a stretch of a read on the edt line by holding down the middle mouse button and dragging the cursor over some bases. They will turn yellow as you drag. Then release the mouse button. A window will pop up giving you some choices of what to do with those (yellow) bases.:

```
      Make High Quality--makes the highlighted bases edited high quality
           (99).  This tells phrap (when it reassembles) that you are
           sure of the sequence here.
      Change Consensus--make the highlighted bases edited high quality and
           change the consensus to agree with that stretch of the read.
           This is a directive to phrap (upon reassembly) to use that
           stretch of that read to be the consensus.
      Make low quality--makes the highlighted bases edited low quality.
           This tells phrap (when it reassembles) that you are not sure
           of the bases here and phrap can go ahead and make a join even
           if the bases in this region don't match perfectly.
      Make Low Quality to Left End--same as above, but all the way to
           the left end of the read.
      Make Low Quality to Right End--same as above, but all the way to
           the right end of the read.
      Change to n's--Change the highlighted bases to n's which means
           they are unknown bases.  This tells phrap (when it
           reassembles) to not make any join based on these bases.  It is
           useful when you believe the bases may be in the chimeric
           portion of a read.
      Change to n's to left--same as above but to left end.
      Change to n's to right--same as above but to right end.
      Change to x's to left--Change the highlighted bases to x's which
           means they are vector.  This tells phrap to ignore these bases
           for the purpose of determining overlap.
      Change to x's to right--same as above but to right end.
      Add Tag--allows user to add any tag to a stretch of read bases.
      Dismiss--you decided you don't really want to do anything with
           this stretch of bases.
```

This popup is made so that nothing else works until you choose something. Try each of these choices, except for tags, which you'll try below.

'Change Consensus' has an additional function--if a read extends out on the right beyond the end of the consensus, you can extend the consensus by using this function. You might want to do this, for example, if crossmatch did not correctly find the cloning site and thus clipped too much. You can add these bases to the consensus by using 'Change Consensus'. Typically, the quality of these bases in the read and in the consensus is 99. That is so that next time phrap runs, it will correctly extend the consensus.

However, if you aren't going to reassemble, you might want to just leave the quality values the way phred originally called them. You can do this by using a Consed parameter (consed.extendConsensusWithHighQuality), which you will learn more about later (see CONSED CUSTOMIZATION).

43) To delete a base, overstrike it with a '*' character. (Phrap ignores '*', so this is the same as deleting the character.) If you overstrike all bases in a column with * characters so the entire column consists of *'s (including the consensus base), there is no way to remove the column. This is OK since when you export the consensus (try the exercise on EXPORTING THE CONSENSUS), the *'s are not exported. While you are editing in Consed, we believe there should be a visual indication that a base was deleted.

SAVING THE ASSEMBLY

44) To save the assembly, pull down the 'File' menu on the Aligned Reads Window, and release on 'Save assembly'. A box will pop up with a suggested name. I suggest you always use the one it suggests. The idea is that the ace files:

```
(project).fasta.screen.ace.1
(project).fasta.screen.ace.2
(project).fasta.screen.ace.3
(project).fasta.screen.ace.4
(project).fasta.screen.ace.5
```

are in order of how old they are. If you feel you are taking up too much disk space, then start deleting the ace files starting at the oldest. I do not recommend that you overwrite existing ace files. The version numbers just keep growing, and that is not a problem.

EXPORTING THE CONSENSUS

45) Exporting the consensus. Bring the Aligned Reads Window into view again. Hold down the left mouse button on the 'File' menu and release the button on 'Export consensus sequence'. Notice that the consensus will be stored (in this case) in a file called 'Contig1.fasta'. Click 'OK'. There is now a file in your edit_dir directory called 'Contig1.fasta' that has the consensus sequence in it. If you want to see the file, bring up another Xterm (if you are UNIX literate), and type:

```
 cd standard/edit_dir
 more Contig1.fasta
```

46) Fancier exporting the consensus. Bring the Aligned Reads Window into view again. Hold down the left mouse button on the 'File' menu but this time release on 'Export consensus sequence (with options)...'. Just export a little snip of the consensus, from 400 to 410. (You will notice this contains a pad * character.) Under "Write Both Bases File and Qual File or Just Bases File?" click "Both Files" Click 'OK'. Consed will want to call this file 'Contig1.fasta' again. You can overwrite the existing file.

Look in your other Xterm at these files:

```
more Contig1.fasta
more Contig1.fasta.qual
```

The one file contains the bases (but no * pads) and the other contains the corresponding qualities of those bases.

47) Exporting the consensus of all contigs at once: Go to the Main Consed Window. Point to 'File', hold down the left mouse button, and release on 'Write all contigs to fasta file'. You then can choose a filename for all contigs to be written to. (In this project there is only 1 contig, so there is no difference between this option and just exporting a contig at a time.)

48) COMPLEMENTING THE CONTIG

Push 'Compl Cont' in the Aligned Reads Window to complement the contig. This displays the opposite strand of the contig including the consensus and all reads. Push this button again to uncomplement it.

49) COLOR MEANS EDITED AND TAGS

(For this step, first click on the 'Dim' menu and release on 'Dim Nothing'.) Point to the 'Color' menu, hold down the left mouse button and release on 'Color Means Edited and Tags'. Notice that the bases that you have edited (make sure you have edited some bases) will stand out in either white or grey (depending on whether the base was made high quality or low quality). Observe this both in the Trace Window and the Aligned Reads window. This colormode is useful if you are interested in easily spotting which bases are edited.

Return to the 'Color Means Quality and Tags' colormode by the following: point to the 'Color' menu, hold down the left mouse button and release on 'Color Means Quality and Tags'.

FIND MAIN WINDOW

50) On the Aligned Reads window, click on 'Find Main Win'. This will cause the Consed Main Window to pop up in the event you have buried it under other windows or iconified it. (This may not work with some settings of your X emulator. In that case you will have to find and click on the Main Window to bring it up.)

MULTIPLE UNDO EDIT

51) Now that the Consed Main Window is visible, click the 'Undo Edit...' button. There will be a popup indicating the most recent edit. (If it says "no edits so far", then bring up a trace and make several edits. Then click on 'Undo Edit...' again.) Click 'undo'. Then you will see the edit that was done before that. Click 'undo'. You can continue undoing if you like. You now know how to undo more than one edit. You cannot choose which edits to undo and which to not undo--edits can only be undone in precisely reverse order from the order you made them. Once you save the assembly, you cannot undo prior edits.

SCROLLING TRACES AND ALIGNED READS TOGETHER

52) In the Aligned Reads window, scroll along the contig to a different point. Click the left mouse button on a read whose trace is already up. Notice that the existing trace instantly scrolls to the corresponding location. Now go to the Trace Window and scroll the traces to a new location. Click on the edt line with the left mouse button. You will notice that the Aligned Reads window will instantly scroll to the corresponding location. Thus you can keep the Aligned Reads window and the traces scrolled to the same location.

SHOW ALL TRACES

53) Go to a region where there are lots of reads, say base 1660. Push down the right mouse button and release on 'Display traces for all reads'. You will see all traces displayed in a scrolling window. You can drag the scrollbar on the right down and up to see all the traces. This feature is particularly useful for polymorphism/mutation detection work. This feature was added to work in cooperation with polyphred. (See CONSED-POLYPHRED intereaction below.)

In this Traces Window, point at one of the bases of one of the reads and click with the left mouse button. The base should start blinking in red. Now push the down arrow key on your keyboard. The cursor should move to the next read. Repeatedly type the down arrow key. Eventually the display should scroll so you can continue to see the read the cursor is on. Try the up arrow key as well.

If there are more than 100 traces at a position, you will see those traces in batches of 100 traces. You can use the bottons at the bottom of the Traces Window labelled "prev 100 traces" and "next 100 traces" to move to the previous and next batches of 100 traces.

There is also a button at the top of the Traces Window that changes between "Show All Traves" and "Show Just Good Traces". A "good trace" means a trace that is all of the following:

```
     * it has a base at the cursor location
     * the trace signal is sufficiently good
     * there is no trag on the read such as a dataNeeded tag that is
listed in the resource:
      consed.showAllTracesDoNotShowTraceIfTheseTagsPresent:
```

EXITING CONSED

54) On the Aligned Reads Window, point to 'File' menu, hold down the left button and release on 'Quit Consed'. If it asks you some questions, answer 'Quit Without Saving and Discard .wrk File'.

ASSEMBLY VIEW

55) Consed can show you a bird's eye view of the Assembly using

forward/reverse pair information, sequence match information, read depth, etc. We have a test database which shows its features.

```
Type:
cd assembly_view/edit_dir
(You might need to type "cd ../.." first depending on where you are.)
ls
Restart consed
```

Double click on "assembly_view.fasta.screen.ace.1"

In the Consed Main Window, click on the button "Assembly View" which is near the upper left corner of the window.

You should see 3 grey bars with pink labels "2", "3", and "1". The bars are the contigs: Pink "1" means Contig1, pink "2" means Contig2, etc. Notice the scale on the contigs. This gives the contig position.

READ DEPTH

56) You should see two graphs above the contig bars: one bright green and one dark green. The dark green graph indicates read depth--the depth of the quality 20 (by default) region of reads. Turn off read depth as follows: Click on the button labelled "What to Show". A menu will popup at that location. Click on the "Read Depth" menu item. A box will appear labelled "Show Read Depth". It has a square (a toggle button) with "show read depth" to the right of the toggle button. Click on the toggle button to change it from appearing pushed in to appearing sticking out. Then click on "Apply". The read depth graph should disappear. If you would like, you can try showing read depth for other qualities other than 20.

Note: the read depth is *not* the # of reads that have quality 20 bases or above, although this number is a good approximation. For example, suppose there is a stretch of 300 Q50 bases, and in the middle of that stretch are 5 Q10 bases. Those Q10 bases will be counted toward the Q20 read depth. (In computer science terms, these bases are part of the maximal Q20 read segment.)

FORWARD/REVERSE PAIR DEPTH

A "forward/reverse pair" is a pair of reads from the same subclone template, each of which is primed within the subclone vector, but one is primed on one side of the insert and the other is primed on the other end of the insert. A forward/reverse pair may both be assembled into the same contig, in which case they should point towards each other and be approximately the insert size apart. A forward reverse pair also might be in different contigs on different sides of a gap.

57) The bright green graph is highest around 7000 to 10000 of Contig2 and around 14000 of Contig3. The bright green graph indicates, for each base, the depth of subclone templates that have a consistent forward/reverse pair. A forward/reverse pair is "consistent" if the forward and reverse are pointing towards each other and are not too far away from each other. ("Too far" is defined as 3 or more standard deviations from the mean of the insert size of templates from a particular library.) In other words, the green graph tells for each base, how many consistent forward/reverse pairs have that base between the forward read and the reverse read. This forward/reverse pair depth is not the same as read depth, which is typically much less. Forward/reverse pair depth is important in that it gives a measure of the confidence of the assembly at a base. If the forward/reverse pair depth is close to zero, as it is in Contig1 position about 9300, there is a likelihood that phrap has made an incorrect join. When the forward/reverse pair depth is zero, the green line turns red, as it does on the right end of Contig3.

INCONSISTENT FORWARD/REVERSE PAIRS

58) The red lines connect the right end of Contig3 with the middle of Contig1. These are filtered inconsisent forward/reverse pairs--they are "inconsistent" because they are not consistent (see above) and they are "filtered" in that they have another inconsistent read close by (at both ends) that is inconsistent for the same reason. If two red lines are on top of one another, it is displayed in purple so you know there is more than one there.

This is a good example of a misassembly. There are many many reads at the right end of Contig3 that are paired with reads in the middle of Contig1. Notice that the forward/reverse pair depth of Contig1 is close to zero around base 9300. (You can use the "Zoom In" button to see this in more detail, but when you are done experimenting with the Zoom buttons and the scroll bar, click on "Zoom Orig" for the rest of this exercise.) This is where phrap made a bad join. If you tear the contig apart there, complement the left part of Contig1, and then join it to the right end of Contig3, the forward/reverse pairs will change from inconsistent to consistent. You will learn later how to do that.

59) Point to one of the red lines. You will notice that it turns yellow. the box near the bottom of the screen tells you a little more about what you have "highlighted" (turned yellow). If you want more information, click with the left mouse button. A window "Clicked Forward/Reverse Pairs" will appear giving information about each highlighted read. Try this. In the "Clicked Forward/Reverse Pairs" Window double click on one of the reads. The Aligned Reads Window should appear with the cursor on that read. This shows how to go from the Assembly View Window to the Aligned Reads Window.

60) You can also go from the Aligned Reads Window to the Assembly View Window. First you must make sure the Assembly View Window is already open (or else open it by clicking on Assembly View in the Consed Main Window). In the Aligned Reads Window, point to a read name, hold down the right mouse button, and release on "Find Read in Assembly View" (one of the last items in the menu the appears when you push down with the right mouse button). If the read is from a subclone that has a forward/reverse pair in the assembly, then the same "Clicked Forward/Reverse Pairs" Window will appear. It will contain not only the read that you pointed to, but all of the other reads from the same subclone as the one you pointed to. In the Assembly View Window, all

of these reads will blink yellow. You can use this procedure to go within the Aligned Reads Window from forward read to reverse read or visa versa.

61) Notice the aqua and purple lines that connect the right end of Contig2 to the left end of Contig3. These are consistent gap-spanning forward/reverse pairs. If there is more than one pair on top of each other, the color is purple. These are the reads that tell you (and Consed, Autofinish, and Phrap) that the right end of Contig2 is connected to the left end of Contig3. As above, point to one to highlight it and click on it to see more information.

62) You can see much more information by clicking on the "What to Show" button, and then when the menu pops up, click on the "Fwd/Rev Pairs" menu item. Up will pop the "Which Fwd/Rev Pairs to Show in Assembly View" Window. Click on "All" next to "Show Inconsistent Forward/Reverse Pairs". Then click "Apply" at the bottom of this window. In this particular example, you just see a few more stray red lines. In a real example, you would probably see so many red lines that it would be a mess. In most cases those inconsistent forward/reverse pairs would be just caused by some laboratory problem (turning a plate around, mislabelling, etc) and not to any misassembly. Thus I suggest that you only generally leave "Show Inconsistent Forward/Reverse Pairs" to "Filtered".

63) Still in the "Which Fwd/Rev Pairs to Show in Assembly View" Window, click on "Show each consistent fwd/rev pair within contigs" (so the button looks as though it is pushed in) and click "Apply". This will show a blue (or purple if there is more than one at a location) square for each consistent forward/reverse pair within a contig. The horizontal position of the square is the center of the subclone (midway between the forward and reverse read) and the vertical position of the square indicates the size of the subclone (higher means a larger subclone). If you really want to see the position of the forward and reverse reads, you can do that too: Click on "Show legs on squares for consistent fwd/rev pairs" ("Show each consistent fwd/rev pair within contigs" must be still on) and click "Apply". What a mess! I believe most of this information is much more easily understood by just showing the "consistent fwd/rev pair depth" (the bright green graph described above). But it is your choice. When you want to highlight a consistent fwd/rev pair, you must point to the square--not the legs. Try it so you understand.

64) Suppose you have an assembly and there are some forward/reverse pairs that you specifically do not want to see in the Assembly View Window. For example, perhaps they are from a plate that was misnamed (or turned around) or from a library that is somehow less reliable. By hiding these forward/reverse pairs, the more reliable/important ones can more easily be seen. This is how you can do that:

In the "Which Fwd/Rev Pairs to Show in Assembly View" Window, notice the line that says:
Do not show templates in file doNotShowInAssemblyView.fof

Underneath this are 3 buttons and probably the one that is selected is "show all templates". Try clicking "do not show specified templates" and click 'Apply'. See if you notice that anything changed in which forward/reverse pairs are displayed. If not, switch back and forth between "show all templates" and "do not show specified templates", each time clicking 'Apply'. When you see a line that appears and disappears, click on it to find what template it is. For example, djs736a2_fp04q146 is one such template. Then from an xterm in the assembly_view/edit_dir directory, type:

```
more doNotShowInAssemblyView.fof
```

You will see the names of the templates that are displayed/hidden.

In order to hide particular forward/reverse pairs, put them into this file. This file can also contain the character '*' which means "match any characters". For example, djs736a1_fp* would match the template

djs736a1_fp04q206

but not

djs736a2_fp01q127

65) Try turning on/off each of the Fwd/Rev Pair options so you understand them. (In this example, there are no "consistent fwd/rev pairs between different scaffolds.")

SEQUENCE MATCHES

66) Notice the curvy orange lines connecting Contig1 with Contig2 and Contig3. These show sequence matches. Point at the one connecting Contig1 and Contig2 and click on it. A "Sequence Matches" box will popup saying that this match has 119 bases and has a similarity of 90.8%. Click on that line so its background turns black. Then click on the button "Show Alignment". Up will pop the Compare Contigs Window with the alignment shown in the lower half of this box. You will learn more about this later (see "JOIN CONTIGS"). For now, dismiss this window.

67) In the Assembly View Window, click on "What to Show" and then when the menu pops up, click on "Sequence Matches". In the "Which Sequence Matches to Show in Assembly View" Window, try clicking off "ok to show sequence matches between contigs". Then click the "Apply" button. You should see the orange lines disappear. (Any highlighted lines will not disappear.) Click "ok to show sequence matches between contigs" back on, and click "Apply" and the lines should be back.

68) Also in the "Which Sequence Matches to Show in Assembly View" Window, change the minimum similarity from 90 to 85. Click "Apply". You should see a lot more orange curvy lines, and now you should also see black curvy lines. If you look carefully, you will see that 2 lines within each pair of orange curvy lines do not cross each other but the 2 lines within each pair of black curvy lines do. This is because orange is used to show direct repeats and black is used to

show inverted repeats (relative to the orientation of the contigs in the Assembly View Window).

69) Also in the "Which Sequence Matches to Show in Assembly View" Window, click on "filter seq matches by size" and set the min size to 400 and the max size to some huge number (e.g., 1000000) and click "Apply". You will see just one direct repeat (orange curvy lines) of size 745.

70) Try some of the other ways of filtering the sequence matches on "Which Sequence Matches to Show in Assembly View".

71) You must learn this step if you are going to ever see sequence matches with your own data, so don't skip this step. If you have problems, it is likely that the phred/phrap/consed package has not been installed correctly and you will need help from your system administrator. Exit Consed and look at the files in assembly_view/edit_dir.

Notice there is a file: assembly_view.fasta.screen.ace.1.aview

This is what Consed uses to show sequence matches in the Assembly View Window.

When you use your own data, you will not have this file so you will need to learn how to create it. Hide it from Consed by (in practice you will never do this step--this is just to simulate the .aview file not being there):

```
mv  assembly_view.fasta.screen.ace.1.aview  assembly_view.fasta.screen.ace.1.aview_hide
```

Now restart consed and select ace file
assembly_view.fasta.screen.ace.1

If you are asked if you want to apply edits, click the "No" button.

Click on "Assembly View" in the Consed Main Window.

You will get the error message:

```
"Sequence matches will not be shown in Assembly View because there is
no file
assembly_view.fasta.screen.ace.1.aview
If you want sequence matches to be shown, click on "What to show:
Sequence Matches" and then "run crossmatch"
```

72) RUNNING CROSSMATCH FOR SEQUENCE MATCHES

Just as the instructions (above) say, click on "What to show" and then when the popup menu appears, click on "Sequence Matches" and then when the "Which Sequence Matches to Show In Assembly View" Window comes up, click on the "Run Crossmatch" button.

Watch the action in the xterm. There should be several pages worth of output from crossmatch that scrolls by in the xterm. If you get an error, it is likely that the phred/phrap/consed package is not correctly installed. You (or your system administrator) should track down the problems and correct them.

If you are successful, then 3 orange pairs of curvy lines will appear in the Assembly View Window--the same as you saw in the steps above.

PULLING OUT READS AND RE-ASSEMBLYING THEM (MINIASSEMBLIES)

When the Assembly View Window indicates, using forward-reverse pair information, that there is a misassembly, Consed provides the tools to correct that misassembly: you can first pull out the the misassembled reads from their current contigs into individual contigs, with a single read per contig. Then you can reassemble those new contigs that each contain a single read. Let's do this:

73) In the Assembly View Window move your cursor so that the red and purple forward/reverse pair lines turn yellow. You will be unable to get them all yellow, but get as many as you can. Then click with the left mouse button. A window labelled "Clicked Fwd/Rev Pairs" should appear with a very long list of reads in it (around 53 reads).

74) In the "Clicked Fwd/Rev Pairs" Window, click on the button labelled "Pull out reads". A window labelled "Put Reads into Their Own Contigs" should appear.

75) In the "Put Reads into Their Own Contigs" Window, select all of the reads. You can do that by clicking with the left mouse button on the first read and then scrolling down to the bottom of the list of reads, holding down the shift key and clicking with the left mouse button on the last read. (When a read is selected, its background should be black.) Click on the button "Remove Highlighted Reads". The Assembly View Window will close and reopen after a few seconds and will complain about not being able to show sequence matches. Save the assembly (see "SAVING THE ASSEMBLY" above) and follow the instructions in "RUNNING CROSSMATCH FOR SEQUENCE MATCHES" (above).

The assembly will now probably contain 4 contigs: 2-3-1c in one scaffold and 4 in the other. That is because when the misassembled reads were pulled out of Contig1, it fell into two new contigs: the new contig 1 and contig 4. All of the reads you pulled out have created Contig5, Contig6, ... and approximately Contig58, each of which contain only a single read.

MINIASSEMBLIES

76) On the Consed Main Window, click the button "Miniassembly". A box will popup labelled "Reassemble Some Contigs". On the left part of the box will be all contigs, from Contig1 to about Contig58. Notice that starting with Contig5 will be contigs that contain only a single

read. On the right will be Contig5 through approximately Contig58. You add or delete from the list on the right. For example, to delete Contig5 from the list on the right, click on it, and then click "Clear Highlighted". The right list should now only contain Contig6 through the last contig. Add Contig5 back to the right list by clicking on Contig5 in the left list and then clicking on the button labelled "Move Highlighted to Right". Contig5 will now appear at the bottom of the list on the right.

77) Leave all of these boxes blank: "-minscore", "-minmatch", "-forcelevel", and "other phrap options:". Keep "Put into separate contigs" selected rather than "Disgard from assembly". Click the "Reassemble" button. If you haven't saved the assembly, a box will popup saying "Error You must first save the assembly before making a miniassembly". Follow the instructions you learned above ("SAVING THE ASSEMBLY") to save the assembly. Then click the "Reassemble" button again and watch the action in the xterm. Lots of output from determineReadTypes.perl, phrap, crossmatch will scroll by in the xterm as those programs run. (If they don't, you haven't correctly installed all of the Consed package.)

78) When the miniassembly is complete, a box will popup asking "Would you prefer to discard this miniassembly and reassemble again?" Click the "No" button.

79) On the Consed Main Window, click the "Assembly View" button. Consed will complain about not being able to show Sequence Matches so save the assembly and follow the instructions in "RUNNING CROSSMATCH FOR SEQUENCE MATCHES" (above). In the Assembly View Window in addition to Contig1, Contig2, Contig3, and Contig4, you should see a few more contigs. These are the result of the miniassembly of all those individual reads.

CONTIG ARRANGEMENT--REORDER CONTIGS

Contigs are arranged by Consed into "scaffolds" using forward/reverse pair information. However, you might have some external information (such as digest information) that tells you a different arrangement. You can use Consed to rearrange the contigs. This new arrangement will be preserved even if you reassemble.

80) Exit Consed and then restart Consed.

Double click on "assembly_view.fasta.screen.ace.1"

(If a window pops up saying "There is an edit history file (a .wrk file)...", click the "No" button.)

Click on the "Assembly View" button. You will see two scaffolds: one on the top row with Contig2 and Contig3, and one on the bottom row with just Contig1. Now suppose that you believe that Contig2 and Contig1 are connected together instead of Contig2 and Contig3. To do this:

81) Within the Assembly View Window, click on the "Contig Arrangement" button. Up will pop a menu. Click on "Reorder Contigs". A "Reorder Contigs" Window will pop up. Enter the following information:

Contig: 2 [Right End] connected to Contig: 1 [Left End]

That is, you must enter "2" and "1" in the contig boxes, and you must click on the first "right end" button.

Then click on the "Add and Restart Assembly View" button. A warning box will pop up telling you that you are crazy, because there are 12 forward/reverse pairs as evidence that the scaffold as displayed in the Assembly View Window is already correct. Click on "yes"--that you are sure.

The Assembly View Window will disappear for a second and reappear, with Consed2 and Contig1 connected together, just as you wanted.

CONTIG ORIENTATION

82) Some users want a scaffold oriented a particular way. For example, one user might be working on a particular gene so wants to always view the top strand of that gene. Another user might be finishing a BAC and wants the 5' end of the BAC on the left of the scaffold. Phrap, however, may not respect their wishes and might have contigs complemented from the way the users want to view them. Consed provides a way for the user to indicate his/her desired orientation, and thereafter if phrap complements a contig from that desired orientation, Consed will complement the contig back when Consed starts up.

To demonstrate this, exit Consed and then restart Consed.

Double click on "assembly_view.fasta.screen.ace.1"

In the Consed Main Window, double click on Contig1. You will see read djs736a2_fp02q494.y1 pointing left. But let's suppose that you would rather the Contig be in the other orientation, with read djs736a2_fp02q494.y1 pointing right.

In the Consed Main Window, click on Assembly View. Then click on the button labelled "contig arrangement". When a popup menu comes up, click on "Reorient Contigs". The "Reorient Contigs Window" should come up. Highlight the scaffold labelled "1" under "Select a scaffold". Click on "flip scaffold". Then push the button labelled "Apply and Restart Assembly View". There will be an error box complaining about not being able to show sequence matches. To fix that, save the assembly and follow the instructions in "RUNNING CROSSMATCH FOR SEQUENCE MATCHES" (above). In the Consed Main Window, double click on Contig1 so the Aligned Reads Window comes up. Scroll to the right end. You will notice that djs736a2_fp02q494.y1 is now on the right end pointing right.

What is the difference between doing this and just complementing the contig, which just requires the click of a button? The difference is that complementing the contig will be undone the next time phrap runs, but using this procedure will be permanent, even if phrap complements the contig.

RESTRICTION FRAGMENTS

We'll look at this feature in Assembly View after we've learned how to use the Restriction Fragment Window.

CONSED-POLYPHRED INTERACTION

Polyphred is a program for finding polymorphic sites; it was developed by Debbie Nickerson's group (contact them at http://droog.mbt.washington.edu).

We have a test database, 'polyphred', which has had polyphred run on it already. Polyphred has put a polymorphism tag on each polymorphic site.

If Consed is running, exit it.

Type:

```
cd polyphred/edit_dir
(You might need to first type "cd ../.." depending on where you are.)
ls
```

Restart Consed.

Double click on example2.fasta.screen.ace.1

When Consed comes up, you should see 2 contigs.
Double click on Contig2

In the Aligned Reads Window, push the left mouse button while pointing to the 'Navigate' menu and release on:

'Toggle feature: when navigating to consensus location, pop up all traces (currently off)'

That will turn this feature on.

Now push the left mouse button while pointing to the 'Navigate' menu and release on 'Tags'. Up should pop a list of tag types. Double click on 'polymorphism'. Polyphred has already been run so the consensus is tagged with polymorphism tags at each polymorphic site. Up will pop a window labelled 'Polymorphism Tags' with a list of sites. Click on 'Next'.

If you correctly followed the instructions above, all the traces should pop up at the first polymorphic site. You may want to reposition the traces window to see it better.

Now ignore the original 'Polymorphism Tags' window and instead click on 'Next' in the *traces* window. This will take you to the next polymorphic site. Pretty nice, huh?

Dismiss the Traces Window.

83) ALPHABETICAL ORDERING OF READS

The reads can be ordered in 3 ways:

a) alphabetically
b) first all the top strand reads and then all the bottom strand reads. The top strand reads are then ordered by the left end of the reads. Same with the bottom strand reads.
c) arbitrarily by a user-provided file

Try changing between a) and b). In the Consed Main Window (click on 'Find Main Win' on the Aligned Reads Window if you can't find the Main Consed Window because it is covered up with other windows), pull down the 'Options' menu, and release on 'General Preferences'. Scroll down until you find 'Display reads sorted alphabetically or by strand/left end of read.' Switch it between 'alpha' and 'strand'. Then click 'Apply and Dismiss'. Notice the effect in the Aligned Reads Window. Many polymorphism and mutation detection labs find that alphabetically sorting is most useful, while many genomic sequencing labs find that sorting by strand/left end of read is most useful.

If you want to use a user-provided file, you must learn CONSED CUSTOMIZATION (below) with resources:

```
consed.showReadsInAlignedReadsWindowOrderedByFile: false
consed.showReadsInAlignedReadsWindowOrderedByThisFile: readOrder.txt
```

After you are done playing with these features, exit Consed and go back to the previous database:

```
cd standard/edit_dir
(You might need to first type "cd ../.." depending on where you are.)

ls
```

Restart Consed.
Double click on standard.fasta.screen.ace.1

When it says "There is an edit history file (a .wrk file)...Do you want to apply those edits?", click on "no".

Double click on Contig1 to bring up the Aligned Reads Window again in preparation for the next step.

NAVIGATING

84) In the Aligned Reads window, pull down the Navigate menu and release on 'Low consensus quality'. You will see a list of locations. Move the 'Low consensus quality' window down so you can see the Aligned Reads window.

Repeatedly click on 'Next' until you reach the end of the list. (Low consensus quality means an area in which the bases each have too high probability of being wrong.) This saves you from having to look through large amounts of high quality data trying to find problem areas.

There are 2 'Next' buttons--one on the Aligned Reads Window and one on the Low Consensus Quality Window. You can click on either, but it is probably more convenient to use the 'Next' button on the Aligned Reads Window. Thus you can keep the Aligned Reads Window in front with input focus and keep the Low consensus quality window pushed out of the way.

You may want to click on the 'Save' button in the Low consensus quality Window to save to a file a copy of this list of problem areas as you work through them.

In our experience, this will be the most important navigate list you will use. In fact, finishing partly consists mainly of adding reads and rephrapping until this list is reduced to nothing.

85) Dismiss the Low consensus quality window. Pull down the 'Navigate' menu again and release on 'High quality discrepancies as above, but omitting tagged compressions and G_dropouts'. You will probably notice there are no entries (unless you created some yourself by editing). That is because there are no high quality discrepancies with this dataset. So let's force there to be some by lowering the quality threshold. First, dismiss the High quality discrepancies window.

Click on 'Find Main Win'. In the Consed Main Window, pulldown the 'Options' menu and release on 'General Preferences'. Notice that the default for 'Threshold for High Quality Discrepancy' is 40. Change it to 15 and click 'Apply & Dismiss'.

Then follow the steps above to bring up the High quality discrepancies menu. Now you will see several entries. Click 'next' repeatedly to go successively to the next high quality discrepancy in the Aligned Reads Window.

You can also double click on a particular line in the High quality discrepancies window to go to that location. Alternatively, you can single click on a line and then click the 'Go' button.

Dismiss the High quality discrepancies window.

86) Similarly, try the other navigate lists: Unaligned high quality regions (this list will be empty with this data set), Edits, Regions covered by only 1 strand and only 1 chemistry, and Regions covered by only 1 subclone.

Unaligned high quality regions are regions in which the traces are high quality so there is no question of the bases, but the region differs so much from other reads that phrap has given up trying to align the region with the consensus. This could be due to a chimeric read, or perhaps the read belongs somewhere else.

We believe that regions covered by only 1 subclone should be covered by a 2nd subclone to prevent the possibility of there being a deletion in the single subclone.

There are so many different problem lists that you may forget to check one of them and thus miss a serious problem. Thus we combined them all into a single list. This is the first menu item: 'Low Cons/High Qual Discrep/Single Stranded/Single Subclone/Unaligned High'. We suggest you use this list.

87) Also try navigate by tags by selecting 'tags' under navigate: when the Select Tag Type Window appears, double click on 'compression'. (Note that you can't do anything else until you deal with this window.) This gives a list of a particular tag type in a particular contig.

88) There is also a way of getting a list of a particular tag type in all contigs: Click on 'Find Main Win'. In the Consed Main Window, point to the 'Navigate' menu, hold down the left mouse button, and release on 'Tags in all contigs'. Continue as in the previous step. (Since there is only one contig, this list will not be any different than the corresponding list for Contig1.)

PRIMER-PICKING

89) Go to some location near the right end of the contig, say base 2470. Click with the right mouse button on the consensus and click on either one of the top strand primer choices (either from subclone template or from clone template). Consed will pause a moment, and then there will appear a selection of primers that pass all of Consed's requirements. (If you get an error message, Consed might not have been correctly installed. See INSTALLING CONSED above.) Templates are also chosen for each primer. You may have to scroll the primer list to the right to see the templates. Consed lists these templates in order of quality--all of them will cover the read you want to make.

Double click on one of the primers in the Primers Window. That will cause the Aligned Reads Window to scroll to show that oligo in context. Click on 'Accept Primer'. A comment box will pop up. Enter some comment and click 'OK'. Notice that a yellow oligo tag, with a little red end, is created on the consensus for that primer. The red end points in the direction of the oligo. The tag contains all the information you need to order that oligo and do the reaction--you will learn how to pop it up below under 'tags'.

What is the difference between 'Pick Primer from Subclone Template' and 'Pick Primer from Clone Template'?

There are 3 differences:

A. which vector file the primers are screened against. In the former case, the primer is screened against the file primerSubcloneScreen.seq and in the latter case against the file primerCloneScreen.seq

B. In checking for false matches elsewhere in the assembly, if the template is the whole clone, then Consed must check for false matches in the *entire* assembly, including all other contigs. But if the template is just going to be a subclone, Consed only needs to check elsewhere in that subclone. Actually, to be conservative, Consed checks for false matches +/- the maximum insert size of a subclone.

C. If you are picking primers for subclone template, then the primer picker can also pick the subclone templates. If it doesn't find any suitable subclone template, it will reject the primer. (By default, picking of subclone templates is turned on. If you prefer to pick your own templates, and want Consed's primer picker to be much faster, you can turn it off temporarily or permanently. To turn it off temporarily, go to the Consed Main Window, point to the Options menu, hold down the left mouse button and release on 'Primer Picking Preferences'. Scroll down to 'Pick Subclone Templates for Primers' and click 'False'. Click on 'Apply and Dismiss'. To change this permanently, see CONSED CUSTOMIZATION below. Beware: you must correctly customize determineReadTypes.perl for template picking to work. See INSTALLING CONSED above.)

If you are interested in the details of primer-picking, see the section 'AUTOFINISH AND PRIMER PARAMETERS' (below).

When you are done editing and have saved the assembly and exited Consed, run ace2Oligos.perl (supplied with this distribution--make sure your system administrator installed it) which will extract all the oligos you just created. This is handy for email ordering of oligos.

In the xterm, type:

```
ace2Oligos.perl  standard.fasta.screen.ace.2  oligos.txt
```

where standard.fasta.screen.ace.2 is whatever the name is of the ace file you just saved.

ace2Oligos.perl does not record the comments that the finisher entered when creating the oligo. If you want to record that as well, you could use the script ace2OligosWithComments.perl which was written by a Consed user and thus is found in the "contributions" directory.

90) WHEN CONSED CAN'T FIND AN ACCEPTABLE PRIMER

Sometimes Consed refuses to pick a primer. This is because it has tried every possible primer and rejected it for one reason or another. If you don't understand why it didn't pick a particular primer, you can ask it as follows:

In the Aligned Reads Window, point to the "Misc" menu, hold down the left mouse button and release on "Check Primer". Enter the left and right consensus positions of the primer, check which strand, and whether the primer is to use subclone templates or the whole clone as a template. Consed will tell you all that is wrong with that primer. Try looking at a top strand subclone primer from 2340 to 2360.

91) PICKING PCR PRIMER PAIRS

In the Aligned Reads Window, go to the location where you want to pick the first PCR primer, say base 500. Point to the consensus, hold down the right mouse button and release on "Top Strand PCR Primer". Then scroll to the location where you want to pick the second PCR primer, say base 2200. Point to the consensus, hold down the right mouse button and release on "Bottom Strand PCR Primer". There will be a pause and then there will be a list of PCR primer pairs. Click on the pair you want and click "Accept Pair".

You can modify the parameters for choosing PCR primer pairs by going to the Consed Main Window, pointing to "Options", holding down the left mouse button, and releasing on "Primer Picking Preferences." For example, by default Consed does not display all PCR primer pairs--this would take too long and give you too many. However, you can ask it to show you all such pairs. In the Primer Picking Preferences, scroll down to "Check All PCR Pairs (huge) or Just Sample?" and click on "All". Then click on "Apply and Dismiss". Then pick PCR primers again, as above. Don't be surprised if you get 10,000 or more pairs of primers!

(PCR Primers are screened for: melting temperature and length, the melting temperature of the 2 primers must be sufficiently close to each other, each primers must not stick to itself or to the other primer, no mononucleotide repeats, only ACGT's (no n's or ambiguity codes), and primer pair must not amplify any other location. There are many more details...)

SEARCH FOR STRING

92) Try the 'Search for String' button (left side of the Aligned Reads Window). Type in a string (such as aaaca), and click 'ok'. There should be a list of 'hits'. Double click on one of the hits (or single click on it and click on 'go'.) Notice that the Aligned Reads Window scrolls to that position and has the cursor on the found string. (It might be complemented.)

Dismiss this window. Try this again, only this time in the Search For String Window select 'Search Just Reads'. Then click 'OK'. You will notice there are many more hits. This is because this shows hits in each read, even if they are at the same consensus position.

You can also try the approximate match search for string by clicking on 'Approximate' instead of 'Exact'. The 'Per Cent Mismatch' only applies to the Approximate match search.

COPY AND PASTE

93) In the Aligned Reads Window, swipe some bases by holding down the left mouse button. You should see the bases turn yellow, at least temporarily. Then click the 'Search for String' button. Use the middle mouse button to paste the bases you have just swiped into the 'Query string:' box. Notice that you can swipe bases either from the consensus or from a read.

The search for string is case-insensitive so don't worry about the pasting being upper or lowercase.

CORRECTING FALSE JOINS MADE BY PHRAP

94) Phrap may put several reads together that you believe do not belong together. (For example, you may see several high quality discrepancies between the reads.) If you are sure these reads do not belong together, you can force a subsequent reassembly by phrap to not assemble those reads together. You do this by finding a location where there is a high quality discrepancy. Then click on the read with the right mouse button and release on 'Tell phrap not to overlap reads discrepant at this location'. There are no high quality discrepancies with this dataset so Consed won't let you do this. (Try it and see.) However, when you use your own data, you may get the chance!

It is possible to automate this procedure using AutoEdit (see USING AUTOEDIT).

ADD NEW READS

95) For this to work, your system administrator must have set up everything correctly. (See below in INSTALLING CONSED.) Assuming you have set everything up correctly, you can now experiment with adding reads.

From a UNIX prompt, copy the new chromatograms into the chromat_dir directory:

```
cp ../chromats_to_add/* ../chromat_dir
```

Exit Consed and bring it up again using the original ace file standard.fasta.screen.ace.1

If it asks if you want to apply edits, just say 'no'.

On the Main Window, click on the Add New Reads button. There will appear a list of files ending with .fof. These are files that contain lists of chromatograms. Double click on 'reads_to_add.fof' Then Consed will ask "If a read doesn't align against any existing contig, do you want to have it go into a contig by itself? (otherwise it will just not be put into the assembly)" Users usually prefer to answer "yes". Consed will ask "Do you want to recalculate the consensus quality values where each of the new reads is aligned?" Answer yes or no, but in practice you should generally answer "yes." There should be lots of progress output in the xterm from which you started Consed. When it completes, there will be a Reads Added Window popup with a report of which reads were added. In this case, it should say that 9 reads were successfully added and list them.

If you get an error message, look carefully at the full error message in the xterm to diagnose the problem. Probably there is some mistake in how you installed Consed. See INSTALLING CONSED (above).

TEAR CONTIG

Just so you get the same results as I do, exit Consed and bring it up again using the original ace file

standard.fasta.screen.ace.1

If it asks if you want to apply edits, just say 'no'.

96) When phrap really screws up, you may want to just tear the contig apart in several places and then join the pieces back together in a different way. Let's try it:

Go to location 1500. Point the mouse at the consensus base at 1500 and push the right mouse button down. Release the button on 'Tear Contig at This Consensus Position'. Up will pop a list of reads with 2 little buttons next to them <- and ->. Leave everything as it is and just click 'Do Tear'. (If you want to play around with which reads goes into which contig, do that another time.)

Now you should have 2 Aligned Reads Windows on top of each other. One should contain 'Contig2' and the other 'Contig3'. Dismiss the little window that says 'Tear Complete'.

JOIN CONTIGS

97) Now let's join these 2 contigs back together:

Click on 'Search for String' and type in the following bases:
agctgccatc

Click 'OK'.

Search for string should find 2 locations, one in Contig2 and one in

Contig3:

```
Contig2      (consensus)      1447-1456   (uncomplemented)
Contig3      (consensus)      829-838     (uncomplemented)
```

Double click on the first one. The Aligned Reads Window for Contig2 will scroll to location 1447 and the window will raise up. In that Aligned Reads Window, click on 'Compare Cont'.

Now double click on the 'Contig3' line in the above Search for String results. The Aligned Reads Window for Contig3 will scroll to location 829 and lift up. In that Aligned Reads Window, click on 'Compare Cont'.

Now the Compare Contigs Window should be visible. In the Compare Contigs Window, try scrolling back and forth. You can change the cursors (blinking red), but if you do, please return them to the locations 1447 and 829 for the next step. The cursors 'pin' these bases together when doing an alignment. (The algorithm is a pinned and banded Smith-Waterman alignment.)

Click on Align. Try scrolling the alignment by dragging the thumb in the lower half of the Compare Contigs. An 'X' means there is a discrepancy between the 2 contigs. There is also a 'P' (see if you can find it!) The P indicates the bases that you pinned together.

You will also notice that some bases are lighter and some are darker. This indicates quality just as in the Aligned Reads Window. You will notice that wherever there an is a discrepancy (an 'X') one of the bases is low quality. This is your cue that the discrepancy is just a base calling error rather than indicating that the two contigs really are different but similar locations.

Click with the left mouse button on either contig in the bottom alignment. You will notice that both contigs will have the red blinking cursor in the same position. Click on 'Scroll Both Aligned Reads Windows' and look at the Aligned Reads Windows to see that they scroll to the corresponding positions. You can have traces up for the contigs, and they will scroll as well. Experiment with this. Then click 'Join Contigs'. The 2 previous Aligned Reads Windows will disappear and there will be a new one which has a new contig 'Contig4'. You have made a join!

Scroll left and right. You will notice that many of the reads are highlighted. These are the reads that came from the previous "right" contig. To unhighlight all of these reads at once, point to the "Misc" menu, hold down the left mouse button and release on "Unhighlight All Reads".

It is possible to have more than one Compare Contigs Windows up at a time. This allows you to investigate a repeat that has more than 2 copies.

COMPARE CONTIGS WINDOW AND INVERTED REPEATS

In the above example, we used the Compare Contigs Window to examine a sequence match between two different contigs. It is also possible to use the Compare Contigs Window to examine a sequence match between two copies of a repeat within the same contig, either direct or inverted.

98) To see this, restart Consed:

```
../../consed_(computer type)
Double click on standard.fasta.screen.ace.1
```

When it says "There is an edit history file (a .wrk file)...Do you want to apply those edits?", click on "no".

Double click on Contig1 to bring up the Aligned Reads Window. Go to position 69 (use the "Pos:" box described above). Click the "Compare Cont" button on the Aligned Reads Window. The Compare Contigs Window will popup, but move it aside. Go to position 2035 in the Aligned Reads Window. Click the "Compare Contig" button again on the Aligned Reads Window. In the Compare Contigs Window there are two copies of Contig1--one on top and one on the bottom. Each has a "complement just in this window" button. Click on the bottom one (the one that has position 2035 blinking red). After clicking on it, you should notice that the numbers on the bottom contig are reversed to they decrease to the right--a copy of Contig1 has been reversed and complemented. Now click the "Align" button. Suddenly, you should see the alignment appear in the bottom half of the Compare Contigs Window. You should see bases between 69-78 aligned against the reversed complement of bases from 2026-2035.

This has shown how you explore an inverted repeat. If you wanted to examine a direct repeat, you would use the same method except you wouldn't click on the "complement just in this window" button.

Compare Contigs is one method of exploring joins of contigs that were not made by phrap. Another method is to use the Assembly View Window (above). They are designed to work together: the Assembly View Window gives a high level view of all sequence matches and takes you to the Compare Contigs Window which shows the alignment of a single sequence match and, if the user so desires, makes a join.

REMOVING READS

99) You can remove individual reads and put them into their own contigs. For example, in the Aligned Reads Window, go to location 2000. Point to the read name of read djs74_2664.s1 and hold down the right mouse button. Release on 'Put read djs74_2664.s1 into its own contig.' Presto-chango! The read is put into its own contig and the old contig is redrawn without the read in it. At this point you should save the assembly--you should always save the assembly after removing reads.

100) You can also remove many reads at once.

Look at the Consed Main Window. Click on "Remove Reads". Type into the "File of read names:" box "reads_to_remove.fof" and either push the "Enter" key or click on "Read File". You should see a list of 2 reads:

```
djs74-2231.s1
djs74-3174.s1
```

You can click back and forth between the choices of "Delete Reads from Assembly" and "Just Put Each Read into Its Own Contig". Try each one.

Delete Reads from Assembly means that the read will no longer appear in Consed. When you are using your own data and you really want to remove reads from the assembly, you must also use the UNIX "rm" command to remove the corresponding phd files from phd_dir and the chromatograms from chromat_dir. Otherwise, the next time you run phredPhrap, the reads, like Phoenix, will rise again to become part of the next assembly.

After you have completed this exercise, restart Consed so that you have all the reads in their original locations for the following exercises.

TAGS

101) Bring up a trace for a read (as above). Swipe some bases on the 'edt' line while holding the middle mouse button down. A list of choices will pop up. Select 'Add Tag'. Type in a comment in the box at the bottom, and select 'comment' from the list of tag types. You will now see a blue box both in the Aligned Reads Window and in the Traces Window on that read.

To see the comment, you can just point to it in the Aligned Reads Window and you will see the comment in the lower right hand corner of the Aligned Reads Window. Alternatively, you can click on that blue tag in the Aligned Reads Window with the right mouse button and release on 'Tag: comment Show more info?'. Alternatively, you can click on the blue tag in the Traces Window with the right mouse button.

Try creating some other kinds of tags: again swipe some bases in the Trace Window by selecting a different tag type. You will notice that different tags are in different colors. You can always use the methods above to see what kind of tag it is if you forget what a particular color means.

You can also define your own tag types. See below CREATING CUSTOM TAG TYPES for how to do that.

CREATING LONG TAGS

102) You can create really, really long tags as follows: Just create a short version of the tag as above for where you want the tag to start. Then figure out the consensus position of where you want the tag to end. In the Aligned Reads Window, click on the short tag with the right mouse button and release on 'tag: show more info?' (as above). A Tag Window will appear for that tag. In the Tag Window, simply change the End Unpadded Consensus Position to the place you want it to end. Then click 'OK'. You will now notice that the tag will be as long as you wanted.

CONSENSUS TAGS

103) You can create tags on the consensus in the same way. In the Aligned Reads Window, use the middle mouse button to swipe some bases on the consensus in the Aligned Reads Window. Up will pop a list of tag types. Click on one of them. Try it again somewhere else. Try it with the tag type being 'comment'. In this case, you must enter a comment. Notice the pretty colors! If you forget which tag type a particular color represents, just point at the colored tag with the mouse and the tag type will be displayed at the bottom of the Aligned Reads Window.

104) Try creating some tags that overlap each other. You will notice that the overlapping region will be purple. If you want to know which tags overlap, you can use any of the methods already discussed.

SEARCH FOR READ NAME

105) Restart Consed using the original ace file

```
standard.fasta.screen.ace.1
```

If it asks if you want to apply edits, just say 'no'.

Instead of clicking on a read or contig name, type a read name into the "Find reads containing (*'s allowed):". If you want to look at the location containing read djs74-2689.s1, you can just type "2689" and then push the "Enter" key and Consed will immediately bring up the Aligned Reads Window with the cursor on read djs74-2689.s1. Suppose that there were more than one read that matched? For example, suppose you type: "26" and then push the "Enter" key. This matches 3 reads:

```
djs74-2689.s1
djs74-2679.s1
djs74-2664.s1
```

Try it and see what happens...

Try entering "26*9" and see what happens. What does the "*" mean?

Try using "Find 1st read starting with:". Try typing djs74-2 You will notice that as you type each letter, the first item in the list that matches the letters typed will be highlighted. Experiment with deleting a few letters and typing others. This is a powerful method of quickly getting to the read name you are interested in. When you get to the name in the list, you do not have to type the rest of the name--just type carriage return or else click on 'OK'.

ONLINE DOCUMENTATION

106) On the Aligned Reads Window or on the Consed Main Window, click on the 'Help' menu and release on 'Show Documentation'. You will see this document. You can search for keywords in it. It is also on the web. Go to http://bozeman.mbt.washington.edu/consed/consed.html, and find "complete documentation" near the bottom of the page.

THE .WRK FILE

107) Consed keeps a log of all changes you make to an assembly: adding new reads, putting reads into their own contigs, making joins and tears, adding and removing tags, and changing bases. This log is kept in a file ending with ".wrk". You can use this file to help you remember exactly what you did to an assembly.

108) You should save your edits by pulling open the 'File' menu on the Aligned Reads Window, and releasing on 'Save assembly'.

RESTRICTION DIGEST

109) Restart Consed.

Double click on "standard.fasta.screen.ace.1"

In the Consed Main Window, click the "Digest" button. For the purpose of this exercise, the full pathname of file of vector sequence can refer to any file of sequence in fasta format. However, when you are using it with your own data it should refer to a file that contains the sequence of your cloning vector. For example, if you are sequencing a BAC, it should contain BAC vector. The sequence must start at the vector/insert junction that you used when you ligated the insert.

Click "OK". You will see a comparison of in-silico fragments (those calculated from the sequence) and real fragments (those in fragSizes.txt which supposedly came from a real gel).

* If a band is red, that means that it doesn't match.
* If a band has a "v" on it, that means it is a vector fragment.
* If a band has a "g" on it, that means it is a gap-spanning fragment.

Move the pointer over the fragments, and you will see the fragment sizes appear. Move the pointer to the in-silico fragment with size 2299. Click on it. You will see the fragment on the left size of the window become highlighted. Click on the button labeled "right end" (2nd row from the bottom of the window) and the Aligned Reads Window will pop up, with the cursor on the right end of the fragment.

Click on "show problems" and navigate through the list of problems by clicking on "next". You will notice that the Gel Window is zoomed in. To return to the original zoom, click on "Zoom Original".

Where it says "Select Enzyme:", point to "EcoRV", hold down the left mouse button and release on "HindIII". This is how you change enzymes.

Click on the button labeled "Text Output". This can be saved to a file and printed out.

Dismiss the restriction digest window. On the Consed Main Window, click the "Digest" button again. Notice the file "fragSizes.txt". This is a file of actual gel fragment sizes. If you don't have an actual gel, but rather you want to just make predictions of fragment sizes from the sequence, you can leave this box blank (erase the "fragSizes.txt"). Try that.

fragSizes.txt has the following format:

```
>EcoRV
448
710
1102
1197
-1
>HindIII
448
508
586
735
801
-1
```

where EcoRV and HindIII are enzymes and the numbers below them are the actual fragment sizes. Each enzyme list is terminated by -1.

Consed does its best to try to figure out which end of the clone insert is connected to which end of the vector. However, it sometimes is wrong. If you believe it is wrong, you can click "compl vector" to try connecting the insert to the vector in the opposite orientation and see if that produces better agreement with the actual digest.

RESTRICTION DIGEST AND ASSEMBLY VIEW

110) Go to the assembly_view sample dataset and bring up the Assembly View Window:

```
cd assembly_view/edit_dir
(You might need to type "cd ../.." first depending on where you are.)
ls
Restart consed
```

Double click on "assembly_view.fasta.screen.ace.1"

In the Consed Main Window, click on the button "Assembly View" which is near the upper left corner of the window.

Also on the Consed Main Window, click on Digest. The "Select Enzyme and Contigs" Window should appear with EcoRV and HindIII selected. Click OK. The "Display Digest" Window should appear.

Now look at the Assembly View Window. You will notice blue, green, and red rectangles under the grey contig bars. These rectangles are the in-silico restriction fragments. Point to one of them-- it will turn yellow and information will be displayed in the information box below. Point to one of the EcoRV fragments, hold down the right mouse button, and release on "Goto fragment in digest window". Notice that in the Display Digest Window, the selected fragment is highlighted both on the left side (the text) and in the Gel (right) side.

PROTEIN TRANSLATION AND OPEN READING FRAMES

111) If you would like, you can see the amino acid translation of the consensus in all reading frames. In the Aligned Reads Window, push down the left mouse button on the 'Misc' menu and release on 'Show Top Strand Protein Translation'. Try again but this time release on 'Show Bottom Strand Protein Translation'. Notice that there are 2 characters that are in magenta color. What are those characters? Why are they made in a different color? To not show the protein translation, push down the left mouse button on the 'Misc' menu and release on 'Don't show protein translation'.

112) You can search for open reading frames (a methionine and a stop codon within the same reading frame) within a contig. In the Aligned Reads Window, push the left mouse button on 'Navigate' and release on 'Search for Open Reading Frames'. Notice that the open reading frames are shown for all 6 reading frames and are sorted by length.

ERROR RATE

113) In the Aligned Reads Window is a box (upper right) labelled 'Err/10kb'. This is the estimated error rate for this contig, and it is a good indicator of when you are done (or not done) finishing. In addition, you can find the error rate for a particular region of contig as follows: Point at 'Misc' menu, hold down the left mouse button, pull down and release on 'Show Error Info For Region'. Fill in the boxes for left and right consensus position, click on 'Calculate' and you will be given the error and single subclone data for that region.

RUNNING PHRED and PHRAP

phred and phrap *must* be run via the phredPhrap perl script. If you don't do this, you are on your own. If you run phred on its own, and then you run phrap on its own, you will get an ace file that will not be usable by Consed. After you have run into problems (and you probably will), then do not email us--instead please use the phredPhrap script. To use the phredPhrap script to run phred and phrap:

114) Type:
```
phredPhrap -V
```

It should say:
```
030326
```
(or newer).

If it does not, then you probably have not installed all the perl scripts from the scripts directory, as directed in INSTALLING CONSED.

115) Make a copy of the standard dataset. E.g.,

(First go up by typing "cd .." until you see "standard" when you type "ls".

Then type:
```
cp -r standard test
cd test
```

116) Delete all the files in phd_dir and edit_dir:

```
rm phd_dir/*
rm edit_dir/*
```

117) cd edit_dir

118) Run phredPhrap by typing

```
phredPhrap
```

That's it--you no longer need to type *any* arguments, and generally you should not. If you want to add phrap options, you can do that:

e.g.,

```
phredPhrap -forcelevel 3
```

Then run Consed on the resulting ace file as indicated in the beginning of the Quick Tour (above). If you have any problems, this is the time to diagnose them before you use your own data.

COMMON PROBLEMS RUNNING PHREDPHRAP

119) Problems were due to polyphred. To check this, in phredPhrap, leave the following line:

```
$bUsingPolyPhred = 0;
```

This will make polyphred not be used. If the problem then goes away, you will know the problem has something to do with polyphred so do not contact any of the phred/phrap/Consed people. Instead, contact the polyphred people: http://droog.mbt.washington.edu and dpc@u.washington.edu and debnick@u.washington.edu

120) Permission problems. Check that you have write access to the phd_dir and edit_dir directories. You can do this by trying to create a file in those directories:

```
touch ../phd_dir/xxx
```
which creates a file

```
ls -l ../phd_dir/xxx
```
which checks if the file was created.

Do the same with ../edit_dir/xxx

If you get a permission problem, do not contact me. UNIX permission problems are very simple for anyone who knows UNIX--get someone locally who understands UNIX and can help you solve the permission problem.

WHAT IS AUTOFINISH?

Autofinish automatically chooses reads for finishing. Autofinish sometimes is able to completely finish a project with no human decisions. In other cases Autofinish mostly finishes a project, and a human just needs to do the final difficult problems since all the routine problems have already been completed by Autofinish. Thus a human finisher is able to complete far more projects in the same length of time.

Autofinish is flexible to the finishing strategy of your lab. It can be used to finish with just universal primer reads, just oligo walks, just minilibraries, or a combination of these. It can be used to finish either genomic or cDNA.

Autofinish will do the following:

-close gaps
-improve sequence quality
-determine the relative orientation of contigs
-ensure that, at each consensus base, at least 2 reads from different templates are aligned

(You can configure Autofinish to do any combination of these tasks.)

Autofinish will suggest the following types of experiments:

-universal primer reads (forward or reverse)
-custom primer reads with subclone templates
-custom primer reads with whole clone templates
-minilibraries (transposon or shatter) from subclone templates
-PCR

(You can configure Autofinish to suggestion any combination of these experiments.)

USING AUTOFINISH

Note: Before you use Autofinish on your own data, you must modify determineReadTypes.perl. See INSTALLING CONSED above for information about this.

To do the exercises in this section, it would help to be able to edit a file under UNIX and run a program under UNIX. If you can't do that, have someone teach you. (It will not work to edit a file on Windows and then transfer to UNIX.) Typical editors on UNIX are vi and emacs, but pico is probably the simplest for occasional users. You can find more information on pico from:

http://www.strath.ac.uk/IT/Docs/IntroToUnix/node122.html

You should also learn how to examine a file in UNIX, how to move around the filesystem, etc. If you don't know how to do this, consult:

http://www.washington.edu/computing/unix/startdoc/files.html
and
http://www.washington.edu/computing/unix/startdoc/directories.html

There are also many books about Unix at bookstores.

121) Type:
```
cd autofinish/edit_dir
```
(You might need to first type "cd ../.." depending on where you are.)

122) Try starting Autofinish by typing:

```
../../consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

(Note 'consed' above may be 'consed_solaris', 'consed_solaris_intel', 'consed_solaris64', 'consed_alpha', 'consed_hp', 'consed_sgi', 'consed_ibm', 'consed_mac', 'consed_linux2.4', 'consed_linux2.6', 'consed_linux2.6_dyn', 'consed_amd64', or 'consed_amd64_dyn', depending on your executable. If you have trouble, use that 'ls' command (see above) or consult the person who installed Consed!)

If Autofinish says:

```
Run-time exception error; current exception: InputDataError
         No handler for exception.
Abort
```

that means that you have not followed the instructions under 'INSTALLING CONSED' above. Please follow those instructions and then try this again.

When you have successfully run the above command, Autofinish will create 7 files:

```
autofinish.fof
(project name).001014.155627.customPrimers
(project name).001014.155627.nav
(project name).001014.155627.out
(project name).001014.155627.sorted
(project name).001014.155627.univForwards
(project name).001014.155627.univReverses
```

Where '001014.155627' is replaced by your current date and time in format YYMMDD.HHMISS. The first file, autofinish.fof, is a file of filenames. It contains the names of the other files.

```
(project name).001014.155627.univForwards
     is the summary file of the suggested universal forward subclone reads
(project name).001014.155627.univReverses
     is the summary file of the suggested universal reverse subclone reads
(project name).001014.155627.customPrimers
     is the summary file of the suggested custom primer reads
```

These are the files you will typically use for directing your bench work. If you like, you can import these files into Excel since the fields are separated by commas.

The .out file is the Autofinish output file. This is the most important file to examine while you are evaluating Autofinish. If you want to know *why* Autofinish picked the reads it did, it will tell you. Consult this file before you start complaining about Autofinish's choices. I've had people complain, and then, once they look in the .out file (*not* any of the other files), they learn information that persuades them that Autofinish was correct all along. This is hard to over-emphasize, but I will try to over-emphasize it:

It will tell you lots more, such as the orientation of the contigs. It will also tell you the value of all Autofinish parameters used. If you try to customize one of the parameters, check in the .out file to be sure that Autofinish used the value you intended.

CONSULT THE .out FILE CAREFULLY IF YOU DISAGREE WITH ANY OF AUTOFINISH'S CHOICES!

The .sorted file gives the reads sorted by contig and position. This file is useful if you want to find what reads Autofinish suggested for a particular location. It is *not* useful for understanding *why* Consed chose a particular read. It is deliberately terse to make it useful for automation the ordering of reads.

The .nav file is a custom navigation file (see "CUSTOM NAVIGATION" far below). This file allows a Consed user to just click 'next', 'next', ... to review all of Autofinish's suggestions in context. This is a great way to quickly and easily review all of the reads suggested by Autofinish.

This finishing tool is designed to be run in batch after each assembly. In a high throughput operation, the production people can make these reads without anyone using Consed to examine the assembly interactively. Only when Autofinish cannot help you any longer (generally after 3 or more times of running Autofinish, making the reads, and re-assembling), must you bring up Consed graphically and examine the assembly.

We suggest that you write some of your own software to parse the summary files to automatically order primers and reads. The summary files (.customPrimers, .univForwards, .univReverses) will not change much but the .out file may change, so don't try to parse it.

AUTOFINISH: MINIMUM NUMBER OF ERRORS FIXED PER READ

123) By default, the minimum number of errors fixed by an experiment is 0.02

Human finishers typically look for low consensus quality regions--regions that have one or more bases below a particular quality threshold. However, Autofinish can do better: it can find regions where the *total* number of errors is greater than some particular cutoff value. This method can find regions where none of the bases are low quality, but many are medium quality and thus the total number of errors in the region is high. Autofinish will also ignore regions that have a very few low quality bases, as long as the total number of errors is smaller than your cutoff. This is a better critereon because it is the total number of errors that you are trying to reduce when finishing--not the number of bases with quality below some arbitrary cutoff.

Two bases of quality 20 have 0.02 errors (on average). Similarly, 20 bases of quality 30 have 0.02 errors (on average). (Quality values were explained at the beginning of this document.) Suppose that you want Autofinish to suggest an additional read for an area that even just has one quality 20 base. (Be aware that Autofinish will consider 10 quality 30 bases to be just as severe as 1 quality 20 base since, on average, they will both have precisely the same number of errors: 0.01)

124) EDIT PARAMETERS: HOW TO CHANGE CONSED/AUTOFINISH PARAMETERS

This shows how to change

consed.autoFinishMinNumberOfErrorsFixedByAnExp. To change any other parameter, follow these same instructions replacing consed.autoFinishMinNumberOfErrorsFixedByAnExp with the parameter you want to change.

In the edit_dir directory is a file called ".consedrc" which you will only see if you use "ls -a" instead of just "ls". In that .consedrc, add the following line:

```
consed.autoFinishMinNumberOfErrorsFixedByAnExp:  0.01
```

You can do this using an editor, such as pico, or you can do it with Consed. To do it with Consed, bring up consed as follows:

```
consed  -ace  autofinish.fasta.screen.ace.1
```

On the Consed Main Window, point to the "Options" menu, push down the left mouse button and release on "Edit Consed/Autofinish Parameters". Up will pop the "Edit Parameters" window. Near the top is "consed.autoFinishMinNumberOfErrorsFixedByAnExp". Point and click in the box on the left containing 0.02 just underneath "consed.autoFinishMinNumberOfErrorsFixedByAnExp". After clicking, the box outline should turn bold and the cursor should start blinking. Change the 0.02 to 0.01. Click on "just project" near the bottom of the window. The box containing 0.01 should turn red indicating that it is now different than the default. Then click "save". A box titled "Name of parameter file to write" should pop up. Click "ok". Note: you can changed more than one of these values before clicking "save".

When using the Edit Parameters Window, I suggest that you do not click on the up and down arrows of the vertical scrollbar because these will scroll by too much. Instead, I suggest you point to the thumb of the vertical slider, hold down the left mouse button and drag the thumb. Alternatively, point to the black space above or below the thumb and click with the left mouse button. You need to try this to understand.

To be sure that everything happened correctly, look at .consedrc file. It should contain the line:

```
consed.autoFinishMinNumberOfErrorsFixedByAnExp:  0.01
```

(If you don't know how to view a file, get a UNIX book and learn the commands "less", "more", "pico", "vi", or "emacs".)

(Get in the habit of checking .consedrc
after using Consed's Edit Parameter Window.)

AUTOFINISH: MINIMUM NUMBER OF ERRORS FIXED PER READ (continued)

Then run Autofinish again:

```
consed  -ace  autofinish.fasta.screen.ace.1  -autofinish
```

Look at the files just created by typing 'ls -tlr' and look at the .out file by bringing it up with your favorite UNIX editor. You should see:

```
PARAMETERS_CHANGED_FROM_DEFAULTS  {
.
consed.autoFinishMinNumberOfErrorsFixedByAnExp:  0.010
.
.
```

Further down is a section:

```
PARAMETERS  {
! If you want to modify any of these parameters, just cut/paste
! the relevant line into your ~/.consedrc file
! (or into the edit_dir/.consedrc file)
! In the following, I have annotated the parameters with the following
! symbols:
!
! (YES)   freely customize to your own site
! (OK)   don't change unless you have a specific need and know what you
!          are doing
! (NO)   don't change this!
```

This section contains all Autofinish parameters, whether you have changed them or not. Thus a changed parameter will be in both lists.

Find
consed.autoFinishMinNumberOfErrorsFixedByAnExp: 0.010
in this second list.

Then compare the .sorted files from this run of Autofinish and the previous run of Autofinish in which the consed.autoFinishMinNumberOfErrorsFixedByAnExp value 0.02 You will notice that there are 2 additional reads suggested when the parameter is 0.01. There is a resequence with dye terminator chemistry of the djs228_474 template and a de novo reverse on template djs228_2632. Look at the .out file to see why Autofinish chose these reads. It will indicate that the first read is mainly to fix 0.01 errors in the region from 2536 to 2545 and the second read to mainly fix 0.01 errors from 969 to 978.

Bring up Consed to see what is in the 10 base region from 2536 to 2545. You will see that there is a quality 25 base at 2539 and a quality 21 base at 2540. After that come some bases whose qualities are in the high 30s.

In the Aligned Reads Window, point at the Misc menu, hold down the left mouse button, and release on Show Error for a Region. Enter 2539 and 2549 for the "Left Consensus Position of Region" and "Right Consensus Position of Region" respectively and click on "Calculate". You will see that there are .0135 errors in this region. This is less than 0.02 so Autofinish will not try to fix this region unless you

```
reduce consed.autoFinishMinNumberOfErrorsFixedByAnExp to 0.01
```

The default is 0.02 because most labs do not want to fix regions that have less than 0.02 errors.

125) DIVERSION: UNIX LESSON

Note for UNIX novices: Earlier, I said that you only needed to know 3 UNIX commands: pwd, ls, and cd. Then I added "ls -a", "less" and an editor (such as pico). Now I want you to learn one more:

```
ls -tlr
```

This is the same as ls, but it puts one file on a line and prints the lines so that the most recent files are on the bottom. Since you will be creating many, many files as you work through these Autofinish exercises, this command gives an easy way to see the files you have just created, without having to always look at autofinish.fof to look for the names of the files you just created.

AUTOFINISH: CHANGING MELTING TEMPERATURES

126) Use 'ls -tlr' to find the most recent .out file. Search in the .out file (using your favorite editor) for MeltingTemp and you will find the following lines:

```
consed.primersMinMeltingTemp: 55
consed.primersMaxMeltingTemp: 60
```

Some labs prefer to use primers with lower melting temperatures. In your .consedrc file, put the following lines:

```
consed.primersMinMeltingTemp: 50
consed.primersMaxMeltingTemp: 55
```

You can do this by following the instructions above under HOW TO CHANGE CONSED/AUTOFINISH PARAMETERS. When you are done doing that, look in the .consedrc file to make sure it contains the above 2 lines.

Then run Autofinish again:

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

Using your favorite editor, check that the .out file you just created says:

```
consed.primersMinMeltingTemp: 50
consed.primersMaxMeltingTemp: 55
```

(You can find the most recent .out file by typing 'ls -tlr'.)

Compare the .sorted files from this run of Autofinish and the previous run. The difference should be the custom primer read:

The previous .sorted file had:
```
tcttttgtctttccatatacatttt,56
```
which means the melting temperature is 56.

The latest .sorted file had:
```
cattttagaatcagtttgttg,50
```
which means the melting temperature is 50.

127) AUTOFINISH: JUST CLOSING GAPS

You could use Autofinish to just close gaps (you are not interested in fixing single subclone regions or weak regions). Add the following to the .consedrc file (and remove everything else so that Autofinish uses the default values for everything else):

```
consed.autoFinishCoverLowConsensusQualityRegions: false
consed.autoFinishCoverSingleSubcloneRegions: false
```

If you are using the Edit Parameter Window to change these values, you will find them when scrolling about 1/3 way down. Change the consed.primersMinMeltingTemp and consed.primersMaxMeltingTemp back to their original values. Then check the .consedrc to make sure it contains the above 2 lines. (Get in the habit of checking .consedrc after using Consed's Edit Parameter Window.)

Now you should see in the .sorted file just 4 reads: one custom primer read pointing out the left end of the contig and 3 reverses off the left end of the contig. The right end is not extended because Autofinish recognizes that it is the end of the BAC.

You can change any of the parameters listed at the top of the Autofinish output file (or actually any of the more exhaustive list of parameters listed in the 'Info' menu, 'Show Consed Parameters' list.)

We believe the defaults are an excellent starting point.

128) AUTOFINISH: JUST CLOSING GAPS JUST USING WALKS

One high-throughput operation was only interested in closing gaps and only interested in using walks to close those gaps. This is the appropriate set of Autofinish parameters to do this:

```
consed.autoFinishCoverSingleSubcloneRegions: false
consed.autoFinishCoverLowConsensusQualityRegions: false
consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads: false
consed.autoFinishAllowPCR: false
consed.autoFinishAllowResequencingReads: false
consed.autoFinishAllowMinilibraries: false
consed.autoFinishNearGapsSuggestEachMissingReadOfReadPairs: false
consed.autoFinishCallReversesToFlankGaps: false
```

(and every other parameter left the default value).

The first 2 parameters are the same as the "AUTOFINISH: JUST CLOSING GAPS" section (above). The other parameters tell Autofinish all of the types of reactions it is not allowed to use, leaving just walks.

Try this. Now you should see only a single read, a walk, pointing left off the left end of the contig.

129) AUTOFINISH: NOT REPEATING FAILED EXPERIMENTS

For this exercise, keep a backup copy of the ace file:

```
cp  autofinish.fasta.screen.ace.1  autofinish.fasta.screen.ace.1.save
```

If you run Autofinish with the -doExperiments parameter (see below), -doExperiments causes Autofinish to record its suggestions in the ace file (hence changing the ace file). If one of these suggested reads fails to fix a problem, when Autofinish is run again it won't pick the same read again.

```
consed -ace (ace file name) -autofinish -doExperiments
```

If a forward or reverse universal primer read failed, Autofinish (when run in a subsequent round) will not suggest that same experiment. If a custom primer read fails, Autofinish will not pick that same experiment again, and it won't pick a custom primer read that is even close to the failed one. 'Close' is defined by the parameter:

```
consed.autoFinishNewCustomPrimerReadThisFarFromOldCustomPrimerRead: 50
```

In addition, Autofinish (the next time it is run) will tell you how well each experiment did in solving the problem it was intended to solve.

Return the parameters to the defaults and try this by running Autofinish twice like this:

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish -doExperiments
consed -ace autofinish.fasta.screen.ace.1 -autofinish -doExperiments
```

and look at the .out file from the 2nd run. (You can find the most recent .out file by typing 'ls -tlr'.) You should see lines such as this:

```
rejecting experiment: reverse universal primer read with template djs228_1094
because an earlier round of autofinish called this with expid: 1
rejecting experiment: reverse universal primer read with template djs228_1422
because an earlier round of autofinish called this with expid: 2
rejecting experiment: reverse universal primer read with template djs228_1034
because an earlier round of autofinish called this with expid: 3
```

This is Autofinish trying experiment after experiment but finding they were already suggested in an earlier round of Autofinish.

You should not type '-doExperiments' if you do not intend to do the experiments Autofinish suggests. If you use -doExperiments, but you don't really do the experiments, and then you run Autofinish again, Autofinish will be very upset--it will think that all of its suggested experiments failed (because it can't find them). It will see that all of the problems are still present but it will think that it should not choose any of those same experiments again so it will suggest different experiments that will not be as good as its original suggestions.

-doExperiments will also cause suggested oligos to be tagged.

Primer id's created by Autofinish use the same naming scheme as primers created in Consed and they will not conflict with each other. For example, if Autofinish creates oligos djs14.1, djs14.2, and djs14.3, then the next primer that a user accepts will be djs14.4. If Autofinish is run a second time, it will start with primer djs14.5.

When you have completed this exercise with -doExperiments, replace the original .ace file by typing:

```
cp  autofinish.fasta.screen.ace.1.save  autofinish.fasta.screen.ace.1
```

130) AUTOFINISH: doNotFinish particular regions

If there is a region that you don't care to finish (e.g., it has already been finished by an overlapping clone or you know there is no gene there), then you can put a doNotFinish tag on the consensus and Autofinish will not try to finish this area.

First, delete the .consedrc file (or, if you are using the Edit Parameter Window of Consed, restore the parameters to their default values) and run Autofinish again:

```
consed  -ace  autofinish.fasta.screen.ace.1  -autofinish
```

Bring up consed:

```
consed  -ace  autofinish.fasta.screen.ace.1
```

and put a doNotFinish tag on the region from 2000 to 4000. (If you don't know how to do that, read through the Consed Quick Tour, above.) Save the assembly as autofinish.fasta.screen.ace.2

Run Autofinish again:

```
consed  -ace  autofinish.fasta.screen.ace.2  -autofinish
```

Look at the .out files for each of the 2 runs of Autofinish. (You can find the most recent .out files by typing 'ls -tlr'.) You will notice in the .out file for the 2nd run of Autofinish that, in the other than the experiments to extend the contig to the left, there is only one experiment which is from 315 to 1662. If you find that experiment in the .out file, it will say "Contig1 0.05 errors fixed in region from 315 to 1662 fixing 0.05 errors from 969 to 978"

The "969 to 978" gives the worst 10 base window that the read is intended to fix. If you look with Consed, you will see that there is a quality 12 base at 974.

You can also use doNotFinish tags to prevent Autofinish from *extending* a contig into a gap by putting a doNotFinish tag near the end of the contig and setting the following Autofinish parameters:

```
consed.autoFinishDoNotExtendContigsWhereTheseTagsAre:  doNotFinish
consed.autoFinishDoNotExtendContigsIfTagsAreThisCloseToContigEnd:  50
```

131) AUTOFINISH: NOT USING PARTICULAR SUBCLONE TEMPLATES

If you no longer have a template that was used in shotgun, and thus you don't want Autofinish to pick that template, you can put it in a file badTemplates.txt in edit_dir. This is a simple file with one name per line.

Using your favorite UNIX editor, create a file called "badTemplates.txt" in edit_dir. Make it contain a single line:

```
djs228_1094
```

Delete .consedrc (or, if you are using the Edit Parameter Window, restore the parameters to their defaults) and run autofinish again:

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

Search the .out file for djs228_1094. You will find one line like this:

```
not using template: djs228_1094 because   in bad templates file
```

Now try deleting badTemplates.txt and running autofinish again the same way. You will notice there are many differences in reads chosen, since djs22_1094 is now available again for making reverses as well as a template for custom primer walks.

badTemplates.txt can accept "*" (match any characters) as part of the name. For example, djs140_23* will eliminate templates:

```
         djs140_235684
          djs140_235783
          djs140_2326
         etc.
```

132) AUTOFINISH: NOT USING ENTIRE LIBRARIES FOR FINISHING

In addition to the badTemplates.txt file, you can use a badLibraries.txt file which contains a list of all libraries that are off-limits to Autofinish (e.g., you threw away all subclone templates from this library or they are from a different lab which gave you the chromatograms but not the templates). Autofinish determines the library of a read by the following in the PHD file:

```
WR{
template dscript 990603:090231
name: djs366_101
lib: library1
}
```

where "library1" is replaced by the actual library name. Take a look at any phd file in autofinish/phd_dir and you will see this. Generally, determineReadTypes.perl puts this library information into the PHD file.

Make sure that badTemplates.txt is deleted and .consedrc is either deleted (or use the Edit Parameter Window to restore the defaults) and run Autofinish again.

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

Now create a file badLibraries.txt containing a single line:

```
lib1
```

and run autofinish again:

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

Look at the .out file. You will see lines like this:

```
not using template: djs228_1034 because   in bad libraries file
not using template: djs228_1051 because   in bad libraries file
not using template: djs228_1094 because   in bad libraries file
.
.
.
```

You will see that there are no reads suggested that use any of these templates, even though some of them (e.g.., djs228_1034) were used in the Autofinish run (above) before you created the badLibraries.txt file.

When you start doing this with your own data, you must put the lib: line into your phd files. Do this by modifying determineReadTypes.perl.

133) MULTIPLE LIBRARIES WITH DIFFERENT INSERT SIZES

If different libraries have different insert sizes, Autofinish must know the insert size of each library. If there are 5 or more forward-reverse pairs, where the forward and reverse are both in the same contig, then Consed/Autofinish calculates the insert size of the library by finding the mean and standard deviation of the insert sizes of these forward-reverse pairs. The maximum insert size of the library is set at the mean plus 2.5 times the standard deviation.

If there are fewer than 5 forward-reverse pairs, where the forward and reverse are both in the same contig, Consed/Autofinish considers this statistical information unreliable so instead relies on a file called 'librariesInfo.txt" which must be placed in edit_dir (where the ace file is). This file looks like this:

```
LIB{
name: lib0
avgInsertSize: 1500
maxInsertSize: 3000
stranded: double
cost: 600.0
}

LIB{
name: lib1
avgInsertSize: 3000
maxInsertSize: 5000
stranded: double
cost: 1000.0
}

LIB{
name: lib2
avgInsertSize: 10000
maxInsertSize: 12000
stranded: double
cost: 5000.0
}
```

'name' is the name of the library. This is the name that goes into the PHD file after the 'lib:' keyword (see AUTOFINISH: NOT USING ENTIRE LIBRARIES FOR FINISHING above). 'avgInsertSize' is the average insert size of the library--the figure to be used by Autofinish if there are not enough forward/reverse pairs for Autofinish to calculate the mean insert size of the library. 'maxInsertSize' is the maximum insert size--if forward/reverse pairs are further apart than this, Autofinish will assume these reads are misassembled. 'stranded' is whether this template is single or double stranded. 'cost' is the cost of making a minilibrary out of a template from this library.

In .consedrc, there must be a line like this:

```
consed.primersMaxInsertSizeOfASubclone: 5000
```

where 5000 is replaced by whatever the maximum insert size of all of your different libraries.

For this exercise make .consedrc have a single line:

```
consed.primersMaxInsertSizeOfASubclone: 12000
```

Alternatively, use the Edit Parameter Window to set consed.primersMaxInsertSizeOfASubclone to 12000.

For this exercise I have a file in edit_dir called "librariesInfo.txt_hide". To make Autofinish pay attention to it, do the following:

```
cp librariesInfo.txt_hide librariesInfo.txt
```

Delete badLibraries.txt:

```
rm badLibraries.txt
```

Before you run Autofinish again, first restart Consed:

```
consed -ace autofinish.fasta.screen.ace.1
```

On Consed's Main Window, point to 'Info', hold down the left mouse button, and release on 'Show Library Info'. You should see the names of your libraries and the correct number of reads in each library. This feature will be useful in debugging your use of librariesInfo.txt

Then run Autofinish again:

```
consed -ace autofinish.fasta.screen.ace.1 -autofinish
```

Look at the .out file. Look for the following:

"Choosing de novo universal primer reads to try to close gaps"

You will see there are many reads under this heading. These are the lib1 and lib2 reads that have a large average insert size and thus span the gap. Autofinish did not choose some of these reads before because, if the insert size were only 1500 bases, these reads would not have helped to close the gap.

When you are done with this exercise, delete librariesInfo.txt and .consedrc

When there are many reads from the same library, Consed/Autofinish will look at the forward/reverse pairs that are within the same contig (so the insert size of that template can be directly measured) and figure out the mean and standard deviation of the insert size of templates from that library. Consed/Autofinish will use these numbers rather than the number from librariesInfo.txt

134) AUTOFINISH CLOSING GAPS WITH MINILIBRARIES

If you wanted Autofinish to *only* suggest minilibraries to close
gaps, use the following parameters:

```
consed.autoFinishAllowWholeCloneReads:  false
consed.autoFinishAllowCustomPrimerSubcloneReads:  false
consed.autoFinishAllowResequencingReads:  false
consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads:  false
consed.autoFinishAllowPCR:  false
consed.autoFinishAllowResequencingAUniversalPrimerAutofinishRead:  false
consed.autoFinishCallReversesToFlankGaps:  false
consed.autoFinishAllowMinilibraries:  true
consed.autoFinishAlwaysCloseGapsUsingMinilibraries:  true
consed.autoFinishPrintMinilibrariesSummaryFile:  true
```

For this exercise, type:

```
cd assembly_view/edit_dir
```
(You might need to first type "cd ../.." depending on where you are.)

Attention! This is *not* the same directory you have been using. It,
autofinish/edit_dir, does not have any gaps so it cannot be used for
this exercise.

Create a .consedrc file with the parameters above in it.
Alternatively, start Consed in this directory and use the Edit
Parameter Window to modify the parameters as above. Then run
Autofinish:

```
consed -ace assembly_view.fasta.screen.ace.1 -autofinish
```

When it has completed, look in the .out file. You will see the
following:

```
Enough existing fwd/rev pairs to establish:
Left end of Contig3 has 13 fwd/rev pairs connecting it to
Right end of Contig2 with gap size -460 (contigs overlap)

Trying to suggest minilibrary for gap between right end of Contig2 and left end of
Contig3
MINILIBRARY{
best template: djs736a2_fp04q274 from lib djs736a2
size: 3607 errors fixed: 0.01 errors fixed per dollar: 0.00
connecting right end of Contig2 to left end of Contig3 with estimated gap size -460

alternative template: djs736a1_fp02q472 from lib djs736a1
size: 1184 errors fixed: 0.01 errors fixed per dollar: 0.00
}
```

You will also see a more terse (but more easily parseable) description
in the .minilibraries file.

The parameter:

```
consed.autoFinishPrintMinilibrariesSummaryFile:  true
```

will cause Autofinish to print a file with name similar to:
(project name).001014.155627.minilibraries

Or you could be more sparing in which gaps you close with
minilibraries and which you do not:

```
consed.autoFinishAlwaysCloseGapsUsingMinilibraries:  false
```

If the parameter above is set to false, then Autofinish will only
choose minilibraries if the gap is the size below or larger:

```
consed.autoFinishSuggestMinilibraryIfGapThisManyBasesOrLarger:  800
```

If you try this in this example, you will see that Autofinish will not
suggest a minilibrary because the gap has negative size (meaning the
contigs overlap) and thus is not more than 800 bases.

Autofinish can suggest more than one minilibrary per gap:

```
consed.autoFinishSuggestThisManyMinilibrariesPerGap:  2
```

is the default, but you can increase it. If you try this, you will
see more alternate templates suggested for the minilibrary.

When you are done, delete the .consedrc file.

135) CLOSING GAPS USING PCR

If you are interested in just closing remaining gaps with PCR, you can
set the following Autofinish parameters:

```
consed.autoFinishAllowWholeCloneReads:  false
consed.autoFinishAllowCustomPrimerSubcloneReads:  false
consed.autoFinishAllowResequencingReads:  false
consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads:  false
consed.autoFinishAllowMinilibraries:  false
consed.autoFinishAllowPCR:  true
consed.autoFinishAllowResequencingAUniversalPrimerAutofinishRead:  false
consed.autoFinishCallReversesToFlankGaps:  false
consed.autoFinishCoverLowConsensusQualityRegions:  false
consed.autoFinishCoverSingleSubcloneRegions:  false
consed.autoFinishNearGapsSuggestEachMissingReadOfReadPairs:  false
```

This will cause Autofinish to try to close all gaps using PCR.

Type:

```
cd assembly_view/edit_dir
```
(You might need to first type "cd ../.." depending on where you are.)

(This is because the autofinish/edit_dir project does not have any
gaps.)

Create a .consedrc file with the parameters above in it. Then run
Autofinish:

```
consed -ace assembly_view.fasta.screen.ace.1 -autofinish
```

When it has completed, look in the .out file. You will see the
following:

```
436 acceptable primers on left end of contig Contig1
503 acceptable primers on right end of contig Contig1
520 acceptable primers on left end of contig Contig2
202 acceptable primers on right end of contig Contig2
523 acceptable primers on left end of contig Contig3
377 acceptable primers on right end of contig Contig3
Please make the following PCR primers:
     ttctgggtctggaggaca 485 to 502 (bottom strand) for left end of Contig1 melt:
 57.5
     taattgggactataggtacatgc 13202 to 13224 (top strand) for right end of Contig1
 melt:  55.1
     tttgttttgttttgtattttgttt 504 to 527 (bottom strand) for left end of Contig2
melt:  55.1
     ctgttctcctgtcattctgg 9950 to 9969 (top strand) for right end of Contig2 melt
:  55.7
     gggcaagagctgtaaagag 114 to 132 (bottom strand) for left end of Contig3 melt:
  55.3
     accaaataacaggtaaaccaaa 15503 to 15524 (top strand) for right end of Contig3
melt:  55.4


Do PCR reactions with the following pairs of primers:
     ttctgggtctggaggaca     tttgttttgttttgtattttgttt
     ttctgggtctggaggaca     ctgttctcctgtcattctgg
     ttctgggtctggaggaca     gggcaagagctgtaaagag
     ttctgggtctggaggaca     accaaataacaggtaaaccaaa
     taattgggactataggtacatgc     tttgttttgttttgtattttgttt
     taattgggactataggtacatgc     ctgttctcctgtcattctgg
     taattgggactataggtacatgc     gggcaagagctgtaaagag
     taattgggactataggtacatgc     accaaataacaggtaaaccaaa
     tttgttttgttttgtattttgttt     gggcaagagctgtaaagag
     tttgttttgttttgtattttgttt     accaaataacaggtaaaccaaa
     ctgttctcctgtcattctgg     gggcaagagctgtaaagag
     ctgttctcctgtcattctgg     accaaataacaggtaaaccaaa
printing experiment summary files:
      assembly_view.020320.130220.univForwards
      assembly_view.020320.130220.univReverses
      assembly_view.020320.130220.customPrimers
      assembly_view.020320.130220.sorted
```

The first is the list of PCR primers to synthesize. The second list
gives which pairs of the primers in the first list to do PCR with.
(It doesn't make sense to do PCR with the primer off the left end of
Contig2 and the primer off the right end of Contig2.)

Some of these pairs of PCR primers will give products and others will
not (or will give enormous products). The ones giving products will
tell you how the contigs are ordered and oriented. You can then
sequence the product to find the gap sequence between the contigs.

136) AUTOFINISH: TOO MANY UNIVERSAL PRIMER READS

St Louis wanted more universal primer reads, so I put in a feature
that allows for redundant universal primer reads. If you get too
many for your taste, then put this into your .consedrc file:

```
consed.autoFinishRedundancy:  1.0
```

The default is 2.0, meaning that Autofinish will try to fix every
problem area twice--once by some universal primer reads and once again
by other universal primer reads. Then, and only then, will it try
oligo walks to finish remaining problems.

Baylor wanted more reverses to close gaps, so I put a feature into
Autofinish that calls *all* reverses near gaps:

```
     (contig)     ________________________

                                          <- reverse 1
                                           <- reverse 2
                                            <- reverse 3
                                            <- reverse 4
```

(including reverses that are likely to fall into the gap) in the hope
that enough of them will hook onto each other that the gap will be
closed. (If there is already a reverse pointing out but no forward,
Autofinish will suggest the forward.) If this feature gives you too
many reverses for your taste, then in your .consedrc file:

```
consed.autoFinishNearGapsSuggestEachMissingReadOfReadPairs:  false
```

137) AUTOFINISH FOR CDNA ASSEMBLIES

The way to use Autofinish for cDNA assemblies is to pretend that the cDNA is a BAC and that you are only going to allow whole clone custom primer BAC reads. To do this, put the following into your .consedrc file:

```
consed.autoFinishAllowResequencingReads:  false
consed.autoFinishAllowWholeCloneReads:  true
consed.autoFinishAllowCustomPrimerSubcloneReads:  false
consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads:  false
consed.autoFinishAllowPCR:  false
consed.autoFinishCDNANotGenomic:  true
consed.autoFinishCheckThatReadsFromTheSameTemplateAreConsistent:  false
consed.checkIfTooManyWalks:  true
consed.autoFinishExcludeContigIfOnlyThisManyReadsOrLess:  0
consed.autoFinishExcludeContigIfDepthOfCoverageOutOfLine:  false
consed.autoFinishExcludeContigIfThisManyBasesOrLess:  0
consed.autoFinishCoverSingleSubcloneRegions:  false
consed.autoFinishContinueEvenThoughReadInfoDoesNotMakeSense:  true
consed.autoFinishCallReversesToFlankGaps:  false
```

You don't want Autofinish to try to extend off the 3' end or the 5' end of the cDNA, right? How is Autofinish going to determine that? It determines it as follows:

In the 5' end read, put the following into the phd file:

```
WR{
primer  determineReadTypes  001019:112654
type: univ fwd
}

WR{
template  determineReadTypes  001019:112654
name: cDNA1
}
```

In the 3' end read (the read that is primed off the polyA tail), put the following into the phd file:

```
WR{
primer  determineReadTypes  001019:112654
type: univ rev
}

WR{
template  determineReadTypes  001019:112654
name: cDNA1
}
```

For all other reads, such as transposon reads and custom primer walks, put the following into the phd file:

```
WR{
primer dscript  001019:112654
type: walk
}

WR{
template  determineReadTypes  001019:112654
name: cDNA2
type: bac
}
```

If you are going to finish many cDNA's, you will find it will work better to modify determineReadTypes.perl than to go editing every phd file.

So Autofinish finds the univ fwd read and assumes it indicates the 5' end of the cDNA and it finds the univ rev read and assumes it indicates the 3' end of the cDNA. (The parameter
consed.autoFinishCDNANotGenomic: true
tells it to try to find the end of the cDNA in this manner.)

There is one additional problem when using Autofinish for cDNA assemblies: initially, the ace file created by phrap is empty since the 3' and 5' reads don't overlap enough. You have *no* contigs for Autofinish to finish. So phrap is of no use initially.

But you can use Consed to create the assembly:

First run phredPhrap to phred both reads and run determineReadTypes.perl Then pick the 3' read and run phd2Ace.perl on it:

```
phd2Ace.perl (name of phd file)
```

This will give you an ace file with one read in it.

Now suppose that you have other reads from the same cDNA. You can use this technique to add them to the ace file:

To add all the reads phrap has neglected to put into the ace file, do the following:

1. create a file of read names. E.g.,

```
djs74_1180.s1
djs74_1432.s1
djs74_1455.s1
djs74_1465.s1
djs74_1532.s1
djs74_1802.s1
djs74_1803.s1
```

Typically, you will get this list of reads by looking in the singlets file.

Then run consed:

```
2. consed -ace old_ace.ace -addNewReads fileOfReadNames.txt -newAceFilename new_ace.ace
```

where:
fileOfReadNames.txt is the name of the file (above) containing
    the read names
new_ace.ace is whatever you want the new ace file to be named
old_ace.ace is the name of the old ace file

Now you have an ace file that contains all the reads you have sequenced for that cDNA. You can now run Autofinish on it:

```
consed -ace new_ace.ace -autofinish
```

138) AUTOFINISH FOR LISTING GAP-SPANNING TEMPLATES

Sometimes people ask me for how to make Autofinish suggest all templates that span a gap. People who ask this question are not using Autofinish to automate finishing--they are using it as a tool in the hand of a human finisher. Although evidence has shown that Autofinish is far more powerful in an automated mode, it is also a powerful tool in the hands of a human finisher. I will specify how to do this, but hope you will move to the next level of using it in an automated manner.

One method is to just shut off Autofinish suggesting any experiments at all:

```
consed.autoFinishCallReversesToFlankGaps:  false
consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads:  false
consed.autoFinishAllowResequencingReads:  false
consed.autoFinishCoverSingleSubcloneRegions:  false
consed.autoFinishCoverLowConsensusQualityRegions:  false
consed.autoFinishNearGapsSuggestEachMissingReadOfReadPairs:  false
consed.autoFinishAllowCustomPrimerSubcloneReads:  false
consed.autoFinishAllowPCR:  false
consed.autoFinishAllowMinilibraries:  false
```

Thus Autofinish will order and orient the contigs, printing out the forward/reverse pairs that connect the contigs, as exemplified below:

```
Examining existing fwd/rev pairs that flank gap at Right end of Contig46

read                          (start   end)   mate read             mate contig (start   end) contig
                              (pos     pos)   name                  name        (pos     pos) length
agroa3_fp19q452.x1u3 ->      -13      824 agroa3_fp19q452.y1    Contig47   ->     365     1282      1844
mag3gpk041f9.y1       ->       29      689 mag3gpk041f9.x1       Contig53   ->  139675  140402    140921
agroa3_fp06q173.x1u3_m ->       91     1053 agroa3_fp06q173.y1    Contig53   ->  139683  140618    140921
mag2gpk013f21.y1      ->      314      887 mag2gpk013f21.x1      Contig57   ->  257472  258018    261664
mag3gpk064f6.x1       ->      542     1173 mag3gpk064f6.y1       Contig53   ->  139061  139709    140921
mag3gpk105a18.y1      ->      710     1318 mag3gpk105a18.x1      Contig53   ->  138774  139329    140921
Enough existing fwd/rev pairs to establish:
Right end of Contig53 has 4 fwd/rev pairs connecting it to
Right end of Contig46 with gap size -17 (contigs overlap)
```

This shows you that (according to the naming convention of this lab), the following templates span the gap between the right end of Contig53 and the right end of Contig46 (clearly one of these contigs is complemented with respect to the other):

```
agroa3_fp19q452
mag3gpk041f9
agroa3_fp06q173
mag2gpk013f21
mag3gpk064f6
mag3gpk105a18
```

However, what are you going to do now with these templates? Walk on them? Resequence the universal primer reads? Whichever you plan to do, why not allow Autofinish to make the suggestions and spend you time on the harder problems?

139) FINISHING A SPECIFIC CONTIG

```
../../consed -ace autofinish.fasta.screen.ace.1 -autofinish -contig Contig1
```

This will just finish Contig1.

140) MARKING THE END OF THE CLONE

Autofinish tries it best to recognize the end of the clone (BAC), and it does pretty well, but you might have information it doesn't have, such as knowing the sequence of the BAC vector or having reads that were primed from within the BAC vector. You can tell Autofinish this information by adding cloneEnd tags. You can do this in Consed as follow:

In the Aligned Reads Window put the cursor on the consensus at the base position marking the end of the insert. Point at the "Misc" menu, hold down the left mouse button and release on "Add Clone End Tag With Insert To Right" (alternatively, "Add Clone End Tag With Insert To Left"). Then save the assembly and run Autofinish.

If you want such tags to be added automatically, you could write a perl script to append such tags to the ace file.

Some sites have found that this is not enough--they want to change all bases beyond the clone end tag to X's. You can do this either interactively or automatically. To do it interactively, in the Aligned Reads Window, put the cursor on the vector base at the vector/insert junction. Hold down the left mouse button on the 'Misc' menu and release the button on either 'Change to X's to Left in All

Reads' or 'Change to X's to Right in All Reads'.

However, if you are using Autofinish, you will probably also want to have this process automated. To do this, set the following parameter in .consedrc:

```
consed.autoEditConvertCloneEndBasesToXs: true
```

(It is set this way by default, so you normally won't need to do this unless you have unset it.)

Then run AutoEdit as follows:

```
consed -ace (name of ace file with the clone end tags) -autoEdit
```

This will create another ace file with a version number one higher than the one you just ran. If you want to specify a particular new ace file name, you can do it this way:

```
consed -ace (old ace file) -autoEdit -newAceFileName (new ace file)
```

After following this procedure, the consensus may start with X's and end with X's like this:

```
XXXXXXXBBBBBBBBBBBBBBBBBBBBBBBBXXXXXXXX
^
position
1
```

If you would rather that the consensus not contain this masked vector, but rather start with base position 1 being the first base of the insert like this below, you must reassemble with phrap.

```
        BBBBBBBBBBBBBBBBBBBBBB
       ^
        position
       1
```

USING AUTOMATED ADD NEW READS

If you are sequencing the same region over and over and you have a reference sequence, phrap may not be a good choice for creating an assembly: phrap will take a long time to run (since many reads match each other), phrap may make several contigs when you know there should be only one, and phrap may not put all the reads into the assembly. Consed provides an alternative to phrap. Use it as follows:

If the reference sequence is in fasta format, first follow the instructions below under ADDING READS WITHOUT CHROMATOGRAM FILES to create a phd file for the reference sequence. Make sure the phd file is in directory phd_dir.

Then from the edit_dir directory, run
phd2Ace.perl (name of phd file)

This will create an ace file for an assembly that just contains the single reference sequence. Run consed to view it and make sure you have followed each of these steps successfully so far.

Make a file (let's call it "reads_to_add.fof") of all the reads you want to add to this assembly. These reads must all be in the directory chromat_dir.

Then run:

```
consed -ace (old ace file) -addNewReads reads_to_add.fof -newAceFilename (new ace file)
```

When this completes, there will be a new ace file with all the reads added.

What Consed does is take each reads and try to align it against the reference sequence. It will thus attempt to make one contig with all of the reads in it. Some reads may not align very well against the reference sequence. In that case, you can tell consed what you want to do by the following parameter in the .consedrc file:

```
consed.addNewReadsPutReadIntoItsOwnContig: ifUnaligned
```

means that if a read does not match the reference sequence very well, it will be put into its own contig. (For information on how to change the .consedrc file, see EDIT PARAMETERS: HOW TO CHANGE CONSED/AUTOFINISH PARAMETERS elsewhere in this document.)

```
consed.addNewReadsPutReadIntoItsOwnContig: never
```

means that if a read does not match the reference sequence very well, it will not be put into the assembly at all.

```
consed.addNewReadsPutReadIntoItsOwnContig: always
```

means that each read is not even compared to the reference sequence, but just put into its own contig.

Consed also will typically recalculate the consensus quality values, unless you specify in your .consedrc file:

```
consed.addNewReadsRecalculateConsensusQuality: false
```

USING AUTOPCRAMPLIFY

If you have a fasta sequence, and you want to amplify part of that sequence using pcr, and you want to select a pair of PCR primers, you can do that using Consed's autoPCRAmplify function. It can handle very high throughput: on a slow computer it takes about 5 minutes to find PCR primers for a hundred different regions.

If the middle region of the sequence is unknown, you can put N's. If you don't know how many N's, put any number--it doesn't matter. For example:

141) For this exercise, create an empty directory and make a fasta file in it called "brian.fasta" with the following contents:

```
>AP000527.C22.6.mRNA.primerRegion 1 70 81 150 smallest
ACAGGGCCCCTCGCGGGCCCTGACGCAGGATGGAGTTGAGGTGGGGGCAG
CGCTGGACCCCAGGGCCCCTNNNNNNNNNNTGCCGCAGTCTTGGATGATG
GGTTCCTAGAAGCTCTCAACATCTCTTCTTAATTGGAGAAAGTGTTAAGC
>AC004019.C22.4.mRNA.primerRegion 1 70 81 150 smallest
AGCTGTGAGCTGTGCAATCATGTAACTAACTTTGTTTAAGTATTGTTTAG
TCTTTCTGGTCTCCAGATGANNNNNNNNNNNTCAGACATTCCACAGCTACC
TAGAGGACATCATCAACTACCGCTGGGAGCTCGAAGAAGGGAAGCCCAAC
```

The numbers 1 70 81 150 means that the left primer should be selected from the region from 1 to 70 of the sequence (starting at 1), so the primer should be chosen from the sequence:

```
ACAGGGCCCCTCGCGGGCCCTGACGCAGGATGGAGTTGAGGTGGGGGCAG
CGCTGGACCCCAGGGCCCCT
```

The 81 150 means that the right primer should be selected from the region from 81 to 150 of the sequence, i.e. from within:

```
TGCCGCAGTCTTGGATGATG
GGTTCCTAGAAGCTCTCAACATCTCTTCTTAATTGGAGAAAGTGTTAAGC
```

"smallest":

```
 --------------------                        -------------------------
                --->                         <---
```

"biggest":

```
 --------------------                        -------------------------
   --->                                                        <---
```

The word "smallest" means that the primers should be chosen so that the product is as small as possible. That means that the left primer should be chosen as far as possible to the right within the 1-70 region and the right primer should be chosen as far as possible to the left within the 81-150 region. If we had instead put "biggest", the primers would instead have been chosen to make the PCR product as large as possible.

Notice that in the diagram above, I didn't make it look like this:

"smallest":

```
 --------------------                        -------------------------
                   --->                    <---
```

(the primers are at the very edge of the regions). The reason is that in general, due to other checks on the primers, the primers that would make the absolute smallest product are not acceptable, and the primers must be backed up. Similarly for "biggest".

142) Then run the following:

```
amplifyTranscripts.perl brian.fasta
```

(The name comes from the fact that this perl program was originally developed to amplify cDNA transcripts.)

You should see about 5 pages of output flash by the screen, ending with:

```
Checking pairs ... Done
Total space allocated: 0.504 Mbytes; currently free: 0.163 Mbytes in 4 blocks
Total space allocated: 0.504 Mbytes; currently free: 0.163 Mbytes in 4 blocks
Total # pairs: 1000, size: 0.044 Mbytes
Total # segment blocks: 1, size: 0.060 Mbytes
Total # diffs: 4, in 2 lists, size: 0.000 Mbytes
```

(This is cross_match output.)

143) Look at the files just created in your directory. In particular, look at for_colleen_unsorted.txt which should look like this:

```
PRIMER_PAIR {
Region: AP000527.C22.6.mRNA.primerRegion   Product size: 67
AP000527.C22.6.mRNA.primerRegionf: AGTTGAGGTGGGGGCAGC temp: 64
AP000527.C22.6.mRNA.primerRegionr: CATCATCCAAGACTGCGGC temp: 63
}


PRIMER_PAIR {
Region: AC004019.C22.4.mRNA.primerRegion   Product size: 59
AC004019.C22.4.mRNA.primerRegionf: TTTAGTCTTTCTGGTCTCCAGATGA temp: 61
AC004019.C22.4.mRNA.primerRegionr: TCTAGGTAGCTGTGGAATGTCTGA temp: 60
}
```

This gives the primers. The top strand primer is the one ending in 'f' and the bottom strand primer is the one ending in 'r'. Both are in 5' to 3' orientation, so the 'r' primer is reverse complemented from the sequence in the original fasta file.

144) To put these primers into 96 well format for ordering, type

```
orderPrimerPairs.perl  no
```

The output will be:

```
to_order.txt (the file in 96 well format)
primersYYMMDD.fasta (a fasta file of the primers)
for_colleen_sorted.txt (a file in the same format as
    for_colleen_unsorted.txt, but sorted by product length)
```

145) Alternatively, autoPCRAmplify can be used directly without the amplifyTranscripts.perl script, but this is a little harder. I suggest skipping this step unless amplifyTranscripts.perl is not meeting your needs. In this case, you must have an ace file readable by Consed.

AutoPCRAmplify requires a file from you that specifies the regions you want to amplify:

Here is an example of how to do this:

```
cd  autofinish/edit_dir
```
(You might need to first type "cd ../.." depending on where you are.)

```
more  autoPCRAmplify.txt
```

You will see such a file:

```
regionA  Contig1  20-100  ->  Contig1  1000-1160  <-  biggest
regionB  Contig1  5200-5270  ->  Contig1  6000-6050  <-  smallest
```

regionA and regionB are the identifiers for the 2 regions that will be amplified.

"Contig1 20-100 ->" says that you want a top strand primer to be within the region 20-100 of Contig1.

"Contig1 1000-1160 <-" says you want the bottom strand primer within the region 1000 to 1160 of Contig1.

"biggest" means that if there are more than one pair of primers that satisfy the above conditions (there will typically be zillions), you want the one with that gives the largest product.   "smallest" means you want the smallest product.

Run autoPCRAmplify by typing:

```
consed  -ace  autofinish.fasta.screen.ace.1  -autoPCRAmplify  autoPCRAmplify.txt
```

Type:

```
ls  -tlr
```

to see what file was just created. It should be a file called

```
autofinish.020301.113327.out
```

(where 020301.113327 is replace by the current date/time in the format YYMMDD.HHMISS)

Look at this file with your favorite UNIX editor and you will see the following (with lots of interspersed information):

```
PRIMER_PAIR  {
id:  regionA  product_size:  1136
tgatttaatataatttcaagaaaatcc  temp:  56  24-50  in  Contig1
cattgtggttttaatttgcatt  temp:  57  1138-1159  in  Contig1
}


PRIMER_PAIR  {
id:  regionB  product_size:  791
ggctgacgcttgtaatcc  temp:  57  5249-5266  in  Contig1
gattatacgcgtgagcca  temp:  56  6022-6039  in  Contig1
}
```

These are the primer pairs for the 2 regions.

USING AUTOEDIT

Autoedit is a program that will read an ace file, make edits, and then write out a new ace file, all without any interaction from the user. Thus Autoedit can be run automatically at night, the same way you can run phredPhrap. Autoedit has various options that are controlled from the .consedrc file the same as the .consedrc file controls Autofinish.

Run AutoEdit as follows:

```
consed -ace (name of exising ace file) -autoEdit
```

This will create another ace file with a version number one higher than the one you just ran. If you want to specify a particular new ace file name, you can do it this way:

```
consed -ace (old ace file) -autoEdit -newAceFileName (new ace file)
```

Autoedit has 3 options:

```
consed.autoEditConvertCloneEndBasesToXs:  true
bool
! If true, will convert to X's bases of all reads that protrude beyond a
! cloneEnd tag.
! (YES)

consed.autoEditTellPhrapNotToOverlapMultiplyDiscrepantReads:  true
bool
! This will find all locations where there are multiple identical
! discrepancies with the consensus (and some other conditions) and try
! to make most of the reads quality 99 at that location so that phrap,
! next time it is run, will not overlap those reads.  This will fix
! many misassemblies.
! (YES)

consed.autoEditTagEditableLowConsensusQualityRegions:  true
bool
! This will find regions that are low quality, but that a human
! finisher could easily determine the correct base and thus
! money could be saved by not having Autofinish suggest additional
! reads overlapping the region
! (YES)
```

ADVANCED PHRAP/CONSED USAGE

146) BACKING OUT EDITS AFTER YOU HAVE SAVED THE ASSEMBLY

If you decide that all your edits are terrible and you want to start over (perhaps you have been training a new finisher), the cleanest solution is to delete everything in phd_dir and edit_dir , but leave everything in chromat_dir and just run
phredPhrap again.

147) SELECTIVELY BACKING OUT EDITS AND REMOVING READS

If you want to back out all edits in just particular reads, I have provided a perl script to do this:

```
revertToUneditedRead (read name)
```

What it does it copy the .phd.1 to 1 greater than the highest version.

Then you must reassemble using the phredPhrap script to create an ace file that has no edits for that particular read. It will have all edits for all other reads.

Why doesn't it just delete all phd files except for the .phd.1? In that case, Consed could not read any previous ace file since all previous versions of ace files would refer to phd files that have been deleted.

148) REMOVING READS FROM AN ASSEMBLY

Create a file containing the filename of all the reads you want to remove, one filename per line.
Then use the perl script

```
removeReads   <file of filenames>
```

Then reassemble using the phredPhrap script.

149) ADDING READS WITHOUT CHROMATOGRAM FILES

This may happen if you, for example, download sequence from Genbank and want to assemble it along with your reads.

There are 2 ways to do this, depending on whether you want to edit the read or not.

a) If you want to edit the read, run mktrace to produce a fake trace. It will have all perfect peaks.

Run:

```
mktrace (name of file with fasta sequence)
```

Then run the phredPhrap script normally. You will be able to bring up the traces in Consed and edit the read.

b) If it is not important to edit the reads, there is a method that is a little faster. Create just a fake phd file using:

```
fasta2Phd.perl (name of file with fasta sequence)
```

It will create a file whose name is taken from the fasta file name: for example, if the fasta filename is Contig1.c.fasta, then the phd file will be called Contig1.c.phd.1 The fasta name in the file is ignored.
You can then put this in the phd_dir, and reassemble using the phredPhrap script.

If the reads are really fake (you don't want the templates to be chosen by Consed/Autofinish as a template for a primer), then the read should end with an extension .c or .a or .c1 or
.c2 ... or .a1 or .a2 or ...   This indicates to Consed/Autofinish that the read is a fake read.

Note: when you are creating phd files such as this, you must start with (read name).phd.1 Do not start with (read name).phd.2 or any higher version number. This is because Consed looks for the .1 version in order to find the original phred calls so it expects there to be a .1 version.

There is also a publicly contributed script "lib2Phd.perl" that takes a fasta file that contains more than one sequence and makes phd files for each of them.

150) ALIGNING READS TO A BACKBONE

If you sequence the same region (in different people or in different species), then you may want them all aligned together, even if phrap doesn't want to put them all together. To align them all together, first use a reference sequence and make an assembly out of it by using mktrace or phd2Phd.perl (see above) followed by phd2Ace.perl (see above). Then add all of the other reads using Consed's Add New Reads feature (either automated or manual--see above).

151) COMPARING READS TO A REFERENCE SEQUENCE

The reference sequence, as in the step above, will just be another read in the assembly. Let's call it "ref". To compare the other reads to it, in the Aligned Reads Window, point at the Navigate Menu, hold down the left mouse button and release on "Compare Reads To Reference Sequence". A Window labelled "Enter Name of Reference Read" will pop up. Enter the name of the read and click "OK". A list of high quality read positions that disagree with the reference read will be displayed.

152) FASTER CONSED STARTUP

You can greatly speed up Consed startup if you are willing to use more disk space. The disk space used will be about equal to the total space used by the PHD files. Try this will a large dataset (you won't notice any difference with the test datasets that come with Consed.)

```
     To use this method of startup:

     1) cd to directory where ace file is kept
     2) type: catPhdFiles.perl
          (This will create a file called phd.ball which is big.)
     3) start consed normally
```

In many situations, this will greatly speed up Consed startup. The amount of speedup depends on which operating system is used: on Linux, the time to read phd files dropped from 75 seconds to 8 seconds, and thus the total time to start up consed dropped from 86 seconds to 17 seconds. I saw similar speedups on Solaris where the phd files are on an nfs mounted disk. However, there was another situation in which the startup time was the same.

Warning: If you create phd.ball as above, Consed will be reading most phd files from phd.ball instead of from ../phd_dir. If you delete phd files in phd_dir, you must also delete phd.ball. Otherwise Consed will give lots of error messages "TIME STAMP MISMATCH" and many things will not work correctly.

153) WHY ARE ALL THE READS NOT IN THE ASSEMBLY?

You will notice that there are some contigs that contain only one read. You will also notice that there are some reads that are not shown by Consed at all, since phrap did not put them into the ace file. Why?

If a read does not have a significant match (with Smith-Waterman score exceeding minscore) to any other read, that read is not included in the ace file. Instead, that read is put in the '.singlets' file. That read will not appear in Consed.

If a read does have a significant match to any other read, then it will appear in the ace file and be shown by Consed. However, such a read might have other problems: it might not be possible to assemble such a read with other reads (in the case of EST's this read may be a unique representative of a particular gene (or a genomic sequence contaminant) that happens to contain an Alu repeat and thus happens to match other reads in the data set; or it may represent the only read of a particular alternatively spliced form; or it may have data anomalies of some sort (chimeras, etc.). Such a read would end up in a contig all of its own.

154) ARE THERE READS THAT ARE TOTALLY UNALIGNED?

Unfortunately, yes. In my opinion, Phrap shouldn't have put them in the assembly at all. But we just have to live with it. You can find if a read is totally unaligned by pointing the the read name in the Aligned Reads Window and holding down the right mouse button. Consed will tell you the aligned positions, the high quality position, and the chemistry of the read.

155) VIEWING THE CHROMATOGRAM OF SINGLETS OR NON-ASSEMBLED READS

If you have a chromatogram, you can use Consed to view it, even if it hasn't been assembled into the ace file. This is common with cDNA assemblies in which the reads don't overlap and thus phrap doesn't put them together into a contig.

To do this, make the same edit_dir, phd_dir,
and chromat_dir as above, put the chromatogram into chromat_dir, run phred on it to generate the phd file which goes into phd_dir.

Then go to edit_dir and run:

```
phd2Ace.perl (name of phd file)
```

For example, if your phd file is myRead.phd.1
from edit_dir, type:

```
phd2Ace.perl  myRead.phd.1
```

This will produce myRead.ace

Then just start Consed normally:
```
consed -ace myRead.ace
```
and you can view the chromatogram.

156) HIDING SOME TYPES OF TAGS

If you have many tags that overlap and thus are purple, you can hide some less relevant tag types so there is less purple and there is less distraction. Make sure you have a few tags visible. Then click on 'Find Main Win'. In the Main Window, open the Options menu, and release on 'Hide Some Tag Types'. A list of tag types will pop up. Select the type that you have visible (above). Then click 'OK'. Go back to the Aligned Reads Window. That tag should still be visible. Click on the button 'Some Tags' in the upper right part of the Aligned Reads Window. Your tag should disappear. The 'Some Tags' button should have changed to 'Sh All Tags'. Click on it again. Your tags should have reappeared.

157) CUSTOM CONTIG NAMES

Normally, when you re-assemble, phrap will name the contigs differently--what was Contig31 before may become Contig32. To help you know which contig is which, Consed allows you to give a name (e.g., "A") to a contig which will persist after re-assembling. To do this, swipe some consensus bases with the middle mouse button (as above). When the "Select Tag Type" box pops up, click on "contigName" and also type a name into the "Contig Name:" field and then click "OK". The next time you re-assemble, the name "A" will appear in the list of contigs on the Consed Main Window.

158) MULTIPLE TRACE POPUP

Bring up dataset standard. In the Aligned Reads window, scroll to a region that has many reads and that has some discrepancies--try position 1162. Hold down the shift key, and click with the middle mouse button on the consensus. At this location 3 traces will pop up--these are the 2 highest quality traces that agree with the consensus (on each strand) and the highest quality trace that disagrees with the consensus. This feature is useful in areas of high coverage when you want to rapidly examine just the most significant traces rather than looking at all of them.

159) MAXIMUM NUMBER OF TRACES DISPLAYED

Bring up dataset standard. Scroll to position 1162. Bring up 4 reads and then try bringing up additional reads.You will notice that new reads are put at the top of the stack of traces and, once there are 4 traces displayed, traces are automatically removed from the bottom of the stack. If you want to change this maximum number of traces to something besides 4, you can do that: In the Consed Main Window (click on 'Find Main Win' on the Aligned Reads window), pull down the 'Options' menu, and release on 'General Preferences'. Try changing the 'Max Number of Traces Shown' to 3. Then click 'Apply and Dismiss'. Now dismiss the Trace Window and again start adding additional traces to the Trace Window. You will notice that now the number of traces shown will not exceed 3.

If you want to view a large number of traces at once, you should use the SHOW ALL TRACES (described above).

160) SCALING THE TRACES

In the Trace Window, grab the thumb of the line that is labelled "V" (for Vertical magnification) and move it back and forth, noticing the effect on the traces. This is useful if the traces are too small or too large. There are several other methods of scaling the traces you will learn later.

161) HOTKEYS FOR EDITING

If you do a lot of editing, you will want to have a faster method of doing these edits than having the popup and selecting an option. Thus the following hot keys exist:

```
     < and > (less than and greater than) to make n's to the left
          and the right (respectively) of the cursor
     control-l and control-r to make low quality to the left and
          the right (respectively) of the cursor
     overstriking with a capital letter (e.g., C instead of c) causes
          the base to become high quality rather than low quality
     overstriking with a lower case letter causes the base to become
         low quality
```

Give these a try.

162) SCROLLING TRACES INDEPENDENTLY

Dismiss all of your Trace Windows. Then pop up traces for 2 different reads in approximately the same location. Scroll one of them. You may want to scroll by clicking the arrows or clicking to the left or right of the thumb. You will notice that both will scroll. Consed will do its best to have corresponding peak lined up. (Consed can't line all of them up because the peak spacing is not uniform and differs from read to read.) Try removing a trace by clicking on one of the 'Remove' buttons in the Trace Window. Try adding other traces. Then click on 'No' for scrolling the traces together and try scrolling. You will now observe that they scroll separately.

163) ABI BASE CALLS

If you want to see the ABI base calls, no problem. Just go to the Consed Main Window. Pull down the 'Options' menu and release on 'General Preferences'. Click on 'True' for 'Show ABI Bases in Trace Window' and then click 'OK' at the bottom of the window. The ABI bases will not be shown immediately--you must first dismiss the trace window and bring it up again. You will then see an additional line with the ABI base calls.

164) MEASURING ERROR RATE AND SINGLE SUBCLONE BASES FOR A REGION

Some contigs have long tails of low quality bases and you would like to find out the error rate for the contig without that long tail. On the Align Reads Window, pull down the Misc menu, and release on 'Show Errors for a Region'. This will tell you both the error rate for the region and the number of single subclone bases for that region.

165) PREVENTING 2 USERS FROM MAKING CONFLICTING EDITS

If there are 2 users that are both editing in the same directory, there is the possibility they will both make edits to the same read. Whoever saves their assembly last will wipe out the edits of the other person, even if they were using different ace files. To help prevent this, consed can warn you if someone else is making edits in the same directory. Set the consed parameter:

```
consed.onlyAllowOneReadWriteConsedAtATime:  true
```

The default is "false" so you have to turn this to true to make it work (see CONSED CUSTOMIZATION).

This will usually work even if the 2 users are on different computers (and the directory is nfs-mounted between them) and even if the different computers have different operating systems. I've tested the following combinations:

```
user 1 on Solaris; user 2 on Solaris
user 1 on Linux; user 2 on Linux
user 1 on Solaris; user 2 on Alpha (Digital Unix)
user 1 on Linux; user 2 on Solaris  <--- does not work
```

Only the last combination doesn't work.

166) PRINTING CONSED WINDOWS

There is a free (or nearly free) program called "xv". One web site is http://www.trilon.com/xv It is written by one of those dying breed of UNIX programmers who just *loved* UNIX and programming and sharing it. His web site is enjoyable because some of his passion comes through. With xv, you can make a postscript file from a Consed window. Then you can print the postscript file on a color printer.

However, since some Consed windows are mostly black (Aligned Reads Window and Traces Window), this uses up a lot of toner and is difficult to read. So go to the Consed Main Window, pulldown the 'Options' menu and release on 'General Preferences'. Scroll down to "Make light background in Aligned Reads Window..." and click on "Do it now". Dismiss any Aligned Reads Windows or Traces Windows and then bring them back up. You will notice the light background. A few other things (traces colors and thickness) are also customized for making color prints.

167) COLOR MEANS MATCH

In the Aligned Reads Window, go to the menu labelled 'color', and pulldown and release on 'color means match'.

Now you notice different colors: The colors have the following meaning:

Blue:	agrees with consensus
Orange:	disagrees with consensus
Yellow:	this stretch of this read was used by phrap to form the consensus
Grey:	Low quality or unaligned ends of reads

NOTE TO LINUX USERS

We have found that there is a large variation among different linux systems (even those with the same kernel) so we have provided 3 different executables in the hope that one will work for you.

Type:

```
uname -a
```

If you see a number starting with 2.4 or 2.5, then use consed_linux2.4

If you see a number starting with 2.6 or higher, then use consed_linux2.6 or consed_linux2.6_dyn. You can try both and see which works best for you. The kind of problems you might have would cause consed to immediately terminate, so if consed comes up at all (you can see the Consed Main Window), that particular executable is fine for you. (See QUICK TOUR OF CONSED for how to start Consed--you must be in the correct directory.)

If you are using consed_linux2.4, in /usr/lib, there must be a file:
libstdc++-libc6.2-2.so.3

If you try to run consed and this is missing, you will see an error
message like this:

```
> ./consed
./consed: error while loading shared libraries: libstdc++-libc6.2-2.so.3: cannot open shared object file: No such file or dir
```

I have provided this file in case you don't have it. Just put it in
/usr/lib and see if that fixes the problem.

One consed user reports:

did a little poking around and found that i needed:
compat-libstdc++-7.3-2.96.118 RPM for i386 since i'm running fedora
core 1 at the moment. ... Anyway, if anyone gets this error tell
them they're missing the Standard C++ libraries for Red Hat 7.3
backwards compatibility compiler and it can be downloaded here:

http://www2.linuxforum.net/RPM/fedora/core/1/Fedora/RPMS/compat-libstdc++-7.3-2.96.118.i386.html

----
NOTE TO ITANIUM LINUX USERS

In /usr/lib, there must be a file: libstdc++-libc6.2-2.so.3

If you try to run consed and this is missing, you will see an error
message like this:

```
> ./consed
./consed: error while loading shared libraries: libstdc++-libc6.2-2.so.3: cannot open shared object file: No such file or dir
```

If you don't have this file already, I have provided it for you in
with the linux consed distribution.

NOTE TO SGI USERS

In /usr/lib, there must be a file: libCsup.so

If you don't have this file, you must get it from SGI. To get it, if
you are on Irix 6.2 through 6.4, request:

SG0001637 'C++ Exception handling patch for 7.00 (and above) compilers
on irix 6.2' (it's on the 'Development Options 7.1' CD).

If you are on Irix 5.3, install patch 1600

To make things easier for you, I've included my libCsup.so
This might save you having to get the patches above.

This is a 64 bit executable so you can use it for large datasets
(over 100,000 reads).

NOTE TO SOLARIS USERS

A.
Do not use /usr/ucb/cc !!! How can you tell if you are using it?
Type:

```
which cc
```

If it says /usr/ucb/cc, you must get gcc or else buy the commercial cc
from Sun (which is /opt/SUNWspro/bin/cc).

If you use /usr/ucb/cc, strange things will happen, including
phd2fasta not working correctly by cutting off the first 2 characters
of read names.

B.
If you are using large files or datasets, please use the executable
consed_solaris64. The other one will not be able to handle files
larger than 2Gb.

NOTE TO MACOSX USERS

If you don't have an X environment already on your MAC, download from
Apple at www.apple.com/software I suggest you use XDarwin in full
screen mode. Use option-apple-A to move back and forth between the
MAC desktop and the X environment.

If you have a 1-button mouse, I've found that:

apple-click = right button click
option-click = middle button click
(some people said the reverse of this)

One Consed user who is experienced in macosx says that XDarwin is not
so friendly and instead suggests running the X11 version found at:

http://www.apple.com/downloads/macosx/apple/x11formacosx.html

or else OroborOSX (http://oroborosx.sourceforge.net/), and a new
(non-beta) version is available
(http://oroborosx.sourceforge.net/download.html).

Please edit the phredPhrap to reflect the correct location of nice
(there is a note in the phredPhrap script about this).

CONSED CUSTOMIZATION

If you want to customize Consed, it would help to be able to edit in UNIX. There is no Microsoft Word in UNIX, but there is emacs, vi, pico and other editors. You must learn one of these if you are going to use the phred/phrap/consed/autofinish system effectively.

You can find more information on pico from:

http://www.strath.ac.uk/IT/Docs/IntroToUnix/node122.html

Point at the 'Info' menu on the Consed Main Window, hold down the left mouse button and release on menu item 'Show Current Consed Parameters'. This shows you what is available to be changed by putting in your ~/.consedrc file.

Point at the 'Info' menu on the Consed Main Window, hold down the left mouse button and release on menu item 'Show Default X Resources'. This shows you what is available to be changed by putting in your ~/.Xdefaults file.

Point at the 'Options' menu on the Consed Main Window, hold down the left mouse button and release on menu item 'Edit Consed/Autofinish Parameters'. This includes most of the parameters found under 'Info/Show Current Consed Parameters' (above). It provides an easy graphical way for you to edit these parameters, if you are not familiar with editing under UNIX. You just change the parameter you want and click "Save". (See HOW TO CHANGE CONSED/AUTOFINISH PARAMETERS (far above)). For the new parameter to take effect, you must restart Consed/Autofinish.

Changes in ~/.consedrc only affect one user. If you want to make a change to affect all Consed users on the system, put a file in some central location (e.g., /usr/local/genome/lib/.consedrc) and then have every user set the environment variable CONSED_PARAMETERS to that the full pathname of the file. For example, if using csh or tcsh, type:

```
setenv  CONSED_PARAMETERS  /usr/local/genome/lib/.consedrc
```

If using bash, type:

```
CONSED_PARAMETERS=/usr/local/genome/lib/.consedrc
export  CONSED_PARAMETERS
```

Anything the user puts in ~/.consedrc will override whatever is in the CONSED_PARAMETERS file.

You can also have different parameters for different projects. Put a .consedrc file in the edit_dir of a particular project. When you are working on that project, whatever is in that .consedrc will override whatever is in your ~/.consedrc file or the CONSED_PARAMETERS file.

CUSTOMIZING NAVIGATE BY SINGLE STRANDED REGIONS AND NAVIGATE BY SINGLE SUBCLONE REGIONS

You can set the parameters:

```
consed.searchFunctionsUseUnalignedEndsOfReads:  false
consed.searchFunctionsUseLowQualityEndsOfReads:  true
```

If you set consed.searchFunctionsUseUnalignedEndsOfReads to be false, then the unaligned ends of a read are not considered to cover the consensus.

If you set consed.searchFunctionsUseLowQualityEndsOfReads to false, then the low quality ends of a read are not considered to cover the consensus.

For example, if the settings are:

```
consed.searchFunctionsUseUnalignedEndsOfReads:  false
consed.searchFunctionsUseLowQualityEndsOfReads:  false
```

then a base in a read is only considered to cover the consensus if it is both in the aligned portion of the read and the high quality portion of the read.

168) .consedrc vs .Xdefaults

Although most Consed parameters now go into .consedrc, there are still a very few that need to stay in .Xdefaults. Here is the rule: if the parameter starts with

```
consed.
```

such as

```
consed.gunzipFullPath:  /bin/uncompress
```

then it goes into .consedrc

If the resource (here it is called a "resource" rather than a "parameter") starts with

```
consed*
```

such as

```
consed*contigwin.background:  Black
```

then it goes in .Xdefaults

169) MAKING LIGHT BACKGROUND FOR SLIDES

Put the following in .Xdefaults:

```
consed*contigwin.background:      black
consed*scaleDrawingArea.background:  grey70
consed*consensusDrawingArea.background:  grey70
consed*editableBasesDrawingArea.background:  grey70
consed*tagsDrawingArea.background:  grey70
consed*phredCalledBasesDrawingArea.background:  grey70
consed*ABICalledBasesDrawingArea.background:  grey70
consed*tracesDrawingArea.background:  grey70
consed*contigwin.background:  grey70
consed*scaleNumbers.foreground:  grey70
consed*scaleGraphics.foreground:  grey70
```

Then the following goes into .consedrc:

```
consed.colorTracesA:  chartreuse
consed.colorTracesC:  royal blue
consed.colorTracesG:  Black
consed.colorScale:  Black
consed.colorScaleBackground:  Grey70
consed.colorSequencingDirectionArrow:  Blue
consed.colorConsensusLabel:  Blue
consed.colorConsensusLabelBackground:  Grey70
```

Then from the command line type
```
xrdb -remove
```
and restart consed

170) COLOR BLINDNESS

One person with Red/Green colorblindness (Deutan), found the following colors helpful:

```
consed.colorTracesG:  Yellow
consed.colorTracesA:  forest green
consed.colorTracesC:  medium blue
consed.colorTracesT:  light coral
```

Put these in a .consedrc in your home directory.

CONSED FOR LARGE ASSEMBLIES OR HUGE ASSEMBLIES

This section only applies to assemblies that have 10,000 reads or more and/or is of a multi-megabase region.

171) To speed up the time for Consed to start up, follow instructions for FASTER CONSED STARTUP (above). Please do this--otherwise you will die of old age waiting for Consed to startup (well, it would take hours anyway). With one bacterial genome assembly with 73,000 reads, using this faster startup method it takes consed 4 minutes to startup--just enough time for coffee.

172) Enough memory is vital with large assemblies. In csh or tcsh type:
```
limit
```
You should see something like this:

```
cputime          unlimited
filesize         unlimited
datasize         2097148 kbytes
stacksize        8192 kbytes
coredumpsize     0 kbytes
vmemoryuse       unlimited
descriptors     64
```

Type:
```
limit datasize unlimited
```
Then type:
```
limit
```
just to see that the number has changed.

173) Make sure you have enough swap space to support the amount of RAM on the computer.

174) Some operating systems and computer hardware is inadequate to support programs requiring large amounts of memory. Our experience is that older versions of Linux will not allow a single process to consume all (or even most) of available memory. More recent versions of linux will allow up to 2Gb, but we haven't seen it allow more. If you need more than 4Gb of memory, you have exceeded the addressing limit of any 32-bit computer, regardless of operating system, and you must use a 64-bit computer: an alpha, Itanium, AMD64, SGI, IBM or Solaris Sparc running in 64 bit mode. Normal PCs are 32-bit computers, and therefore won't work for large datasets.

FOR PROGRAMMERS AND FELLOW TRAVELLERS ONLY

175) COMPRESSING CHROMATOGRAMS

If you are interested in compressing your chromatogram files, go into chromat_dir and gzip one of the chromatogram files. Make sure that gunzip is in /usr/local/bin   (You can change this location via the Consed parameter

```
consed.gunzipFullPath:  /usr/local/bin/gunzip
```

--see CONSED CUSTOMIZATION (above), but it will be easiest for you and your users if you just put gunzip (or a link to it) in /usr/local/bin and not have to bother with Consed parameters.)

Restart Consed and bring up the corresponding trace. You will notice no appreciable delay.

176) READING CHROMATOGRAMS OUT OF AN EXTERNAL DATABASE

Normally, chromatograms are kept in ../chromat_dir. If you want to keep them somewhere else (such as in an external database), you can do that. When the chromatogram is needed (when the user asks to view a trace), Consed will call an external program, passing it the name of the read required, and then look for the chromatogram in /tmp (by default). It will read the chromatogram and then delete it. Use the parameters:

```
consed.alwaysRunProgramToGetChromats:  true
consed.programToRunToGetChromats:  /usr/local/bin/programToGetChromat
```

In this case, "programToGetChromat" is the name of the program that gets the chromatogram and puts it into /tmp.

177) CONSED -ACE

Try bringing up Consed like this:

```
consed -ace (name of ace file)
```

This can be useful if you are going to have Consed brought up from some other program.

178) NO PHD FILES

Try bring up Consed like this:

```
consed  -nophd
```

This mode allows you to view an assembly when you don't have phd files or chromatograms but you only have the ace file. I do not recommend nor support this option! There are so many things that do not work with this option that I haven't bothered to keep track of them, but here are a few items: can't make joins, can't recalculate consensus quality, can't view traces, can't edit, autofinish will not give good results, can't view quality of the read bases, ...

179) CREATING CUSTOM TAG TYPES

You can add your own tag types by creating a file of your custom tag types. The file looks like this:

```
mytag1  red consensus  yes
mytag2  purple  both  yes
mytag3  green  read  no
```

     field 1 ("mytag1") is the tag name
     field 2 ("red") is the color
     field 3 is "consensus", "read", or "both" depending on which kind of tag
         it is
     field 4 is "yes" or "no" depending on whether the user can add
          this tag in Consed (by swiping) or whether it is a tag that
          can only be viewed in Consed (presumably it would be added by
          some software of yours before the user sees it in Consed).

If the file is called "/usr/local/genome/lib/tagTypes.txt", then in .consedrc put the following line:

```
consed.fileOfTagTypes:  /usr/local/genome/lib/tagTypes.txt
```
so that Consed knows where the file is.

Once you have done this, the user of Consed can add tags of these types in the method described in TAGS of the Quick Tour (above).

The list of available colors is found in the file rgb.txt found in /usr/lib/X11/rgb.txt on Linux or /usr/openwin/lib/rgb.txt on Solaris. For more information, consult any X-Windows reference, since this has nothing to do specifically with Consed. For your convenience, here are most of the color names. One way to find out what they look like is to try them:

snow	SlateBlue2	coral1
ghost white	SlateBlue3	coral2
white smoke	SlateBlue4	coral3
gainsboro	RoyalBlue1	coral4
floral white	RoyalBlue2	tomato1
old lace	RoyalBlue3	tomato2
linen	RoyalBlue4	tomato3
antique white	blue1	tomato4
papaya whip	blue2	OrangeRed1
blanched almond	blue3	OrangeRed2
bisque	blue4	OrangeRed3
peach puff	DodgerBlue1	OrangeRed4
navajo white	DodgerBlue2	red1
moccasin	DodgerBlue3	red2
cornsilk	DodgerBlue4	red3
ivory	SteelBlue1	red4
lemon chiffon	SteelBlue2	DeepPink1
seashell	SteelBlue3	DeepPink2
honeydew	SteelBlue4	DeepPink3
mint cream	DeepSkyBlue1	DeepPink4
azure	DeepSkyBlue2	HotPink1
alice blue	DeepSkyBlue3	HotPink2
lavender	DeepSkyBlue4	HotPink3
lavender blush	SkyBlue1	HotPink4

misty rose	SkyBlue2	pink1
white	SkyBlue3	pink2
black	SkyBlue4	pink3
dark slate gray	LightSkyBlue1	pink4
dim gray	LightSkyBlue2	LightPink1
slate gray	LightSkyBlue3	LightPink2
light slate gray	LightSkyBlue4	LightPink3
gray	SlateGray1	LightPink4
light grey	SlateGray2	PaleVioletRed1
midnight blue	SlateGray3	PaleVioletRed2
navy	SlateGray4	PaleVioletRed3
cornflower blue	LightSteelBlue1	PaleVioletRed4
dark slate blue	LightSteelBlue2	maroon1
slate blue	LightSteelBlue3	maroon2
medium slate blue	LightSteelBlue4	maroon3
light slate blue	LightBlue1	maroon4
medium blue	LightBlue2	VioletRed1
royal blue	LightBlue3	VioletRed2
blue	LightBlue4	VioletRed3
dodger blue	LightCyan1	VioletRed4
deep sky blue	LightCyan2	magenta1
sky blue	LightCyan3	magenta2
light sky blue	LightCyan4	magenta3
steel blue	PaleTurquoise1	magenta4
light steel blue	PaleTurquoise2	orchid1
light blue	PaleTurquoise3	orchid2
powder blue	PaleTurquoise4	orchid3
pale turquoise	CadetBlue1	orchid4
dark turquoise	CadetBlue2	plum1
medium turquoise	CadetBlue3	plum2
turquoise	CadetBlue4	plum3
cyan	turquoise1	plum4
light cyan	turquoise2	MediumOrchid1
cadet blue	turquoise3	MediumOrchid2
medium aquamarine	turquoise4	MediumOrchid3
aquamarine	cyan1	MediumOrchid4
dark green	cyan2	DarkOrchid1
dark olive green	cyan3	DarkOrchid2
dark sea green	cyan4	DarkOrchid3
sea green	DarkSlateGray1	DarkOrchid4
medium sea green	DarkSlateGray2	purple1
light sea green	DarkSlateGray3	purple2
pale green	DarkSlateGray4	purple3
spring green	aquamarine1	purple4
lawn green	aquamarine2	MediumPurple1
green	aquamarine3	MediumPurple2
chartreuse	aquamarine4	MediumPurple3
medium spring green	DarkSeaGreen1	MediumPurple4
green yellow	DarkSeaGreen2	thistle1
lime green	DarkSeaGreen3	thistle2
yellow green	DarkSeaGreen4	thistle3
forest green	SeaGreen1	thistle4
olive drab	SeaGreen2	gray0
dark khaki	SeaGreen3	gray1
khaki	SeaGreen4	gray2
pale goldenrod	PaleGreen1	gray3
light goldenrod yellow	PaleGreen2	gray4
light yellow	PaleGreen3	gray5
yellow	PaleGreen4	gray6
gold	SpringGreen1	gray7
light goldenrod	SpringGreen2	gray8
goldenrod	SpringGreen3	gray9
dark goldenrod	SpringGreen4	gray10
rosy brown	green1	gray11
indian red	green2	gray12
saddle brown	green3	gray13
sienna	green4	gray14
peru	chartreuse1	gray15
burlywood	chartreuse2	gray16
beige	chartreuse3	gray17
wheat	chartreuse4	gray18
sandy brown	OliveDrab1	gray19
tan	OliveDrab2	gray20
chocolate	OliveDrab3	gray21
firebrick	OliveDrab4	gray22
brown	DarkOliveGreen1	gray23
dark salmon	DarkOliveGreen2	gray24
salmon	DarkOliveGreen3	gray25
light salmon	DarkOliveGreen4	gray26
orange	khaki1	gray27
dark orange	khaki2	gray28
coral	khaki3	gray29
light coral	khaki4	gray30
tomato	LightGoldenrod1	gray31
orange red	LightGoldenrod2	gray32
red	LightGoldenrod3	gray33
hot pink	LightGoldenrod4	gray34
deep pink	LightYellow1	gray35
pink	LightYellow2	gray36
light pink	LightYellow3	gray37
pale violet red	LightYellow4	gray38
maroon	yellow1	gray39
medium violet red	yellow2	gray40
violet red	yellow3	gray41
magenta	yellow4	gray42
violet	gold1	gray43
plum	gold2	gray44
orchid	gold3	gray45
medium orchid	gold4	gray46
dark orchid	goldenrod1	gray47
dark violet	goldenrod2	gray48
blue violet	goldenrod3	gray49
purple	goldenrod4	gray50
medium purple	DarkGoldenrod1	gray51
thistle	DarkGoldenrod2	gray52
snow1	DarkGoldenrod3	gray53
snow2	DarkGoldenrod4	gray54
snow3	RosyBrown1	gray55
snow4	RosyBrown2	gray56
seashell1	RosyBrown3	gray57

```
seashell2                 RosyBrown4          gray58
seashell3                 IndianRed1          gray59
seashell4                 IndianRed2          gray60
AntiqueWhite1             IndianRed3          gray61
AntiqueWhite2             IndianRed4          gray62
AntiqueWhite3             sienna1             gray63
AntiqueWhite4             sienna2             gray64
bisque1                   sienna3             gray65
bisque2                   sienna4             gray66
bisque3                   burlywood1          gray67
bisque4                   burlywood2          gray68
PeachPuff1                burlywood3          gray69
PeachPuff2                burlywood4          gray70
PeachPuff3                wheat1              gray71
PeachPuff4                wheat2              gray72
NavajoWhite1              wheat3              gray73
NavajoWhite2              wheat4              gray74
NavajoWhite3              tan1                gray75
NavajoWhite4              tan2                gray76
LemonChiffon1             tan3                gray77
LemonChiffon2             tan4                gray78
LemonChiffon3             chocolate1          gray79
LemonChiffon4             chocolate2          gray80
cornsilk1                 chocolate3          gray81
cornsilk2                 chocolate4          gray82
cornsilk3                 firebrick1          gray83
cornsilk4                 firebrick2          gray84
ivory1                    firebrick3          gray85
ivory2                    firebrick4          gray86
ivory3                    brown1              gray87
ivory4                    brown2              gray88
honeydew1                 brown3              gray89
honeydew2                 brown4              gray90
honeydew3                 salmon1             gray91
honeydew4                 salmon2             gray92
LavenderBlush1            salmon3             gray93
LavenderBlush2            salmon4             gray94
LavenderBlush3            LightSalmon1        gray95
LavenderBlush4            LightSalmon2        gray96
MistyRose1                LightSalmon3        gray97
MistyRose2                LightSalmon4        gray98
MistyRose3                orange1             gray99
MistyRose4                orange2             gray100
azure1                    orange3             dark grey
azure2                    orange4             dark blue
azure3                    DarkOrange1         dark cyan
azure4                    DarkOrange2         dark magenta
SlateBlue1                DarkOrange3         dark red
                          DarkOrange4         light green
```

180) ADDING TAGS FROM OTHER PROGRAMS

You can also write external programs that add tags to the ace file and/or the phd files. Both RT (read) and CT (consensus) tags can be appended to the end of the ace file. BEGIN_TAG tags can be appended to the end of the phd files. Do not rewrite the ace file or the phd file--there is no need to do so and it will cause problems.

181) CONTROL OF CONSED FROM SOME OTHER PROGRAM

Consed can be controlled by some other program. For example, you might have a program that displays mapping data and you would like the user to be able to click on a location and have Consed come up showing the bases in that region. This feature allows a programmer to do this.

The external program can start up Consed as follows:

```
consed -socket (local port number) -ace (ace filename)
```

For example,

```
consed -socket 5432 -ace standard.fasta.screen.ace.1
```

After Consed completes coming up (including you clicking whether you want to apply edits), you will see the message in the xterm:

```
success bind to local port number: 5432
```

And then you will see a file created by Consed in the default directory called consedSocketLocalPortNumber

This gives the port number of the Berkeley socket that Consed has opened and is listening on. Thus your program can read this file and create a connection to the Berkeley socket created by Consed.

Once the connection is established, your program can send commands to Consed at that socket indicating to Consed which contig to display and what consensus position to scroll to. Currently, the only acceptable commands are:

```
Scroll (contigname) (consensus position)<return>
PopupTraces (read name) (unpadded read position in the direction of sequencing)<return>
```

'Unpadded read position in the direction of sequencing' is the position from the right end, if the read is a bottom strand read.

Just send such a command to the Berkeley socket, and Consed will respond appropriately. (Currently, Consed doesn't like it if another process establishes a connection and then terminates without first terminating the connection.)

Here is an example: In standard/edit_dir, start consed as follows:

```
consed -socket 5432 -ace standard.fasta.screen.ace.1
```

Then in the same directory, run the following perl script (thanks to
Bill Gilliland):

```
#! /usr/local/bin/perl -w

# open a socket to consed

use IO::Socket;

# Consed makes a file which has nothing but the port number.
# This will enforce running this program in the edit_dir, too!
open(SOCKETFILE, "consedSocketLocalPortNumber")
            || die "Can't find consedSocketLocalPortNumber!\n";
$portNumber = <SOCKETFILE>;
close(SOCKETFILE) || die "Can't close consedSocketLocalPortNumber!\n";

$socket = IO::Socket::INET->new("localhost:$portNumber") or die $@;

print $socket "Scroll Contig1 100\n";
while(<> ) {
     print "waiting for you to type...";
}
```

Consed should scroll to position 100.

182) AUTOMATIC ORDERING OF OLIGOS

I heard of a finisher who manually ordered 72 oligos. She had to cut/paste the bases of each oligo. That is not only painful, but also error prone. I've supplied you a script that you can use to automatically determine which oligos have been newly requested since the last order, aggregate them into a single order, and email the request off.

The script is ace2Oligos.perl. It takes as command line arguments the name of an ace file and the name of the oligo file. The oligo file is a list of oligos that have been ordered for that particular project, and looks like this:

```
name=G1980A181.1
sequence=ctgcatggctaggga
template=seq from subclone
date=980427 temp=52

name=G1980A181.2
sequence=tcttactttctgactttcattt
template=seq from clone
date=980427 temp=50
```

ace2Oligos.perl finds all oligo tags in the ace file and makes sure that all of them are in this oligo file.

To automatically order oligos each night, there is an additional script you will have to write. I suggest that you run your script each night under cron and that it do the following:

for each project, it will look for the most recent ace file. It will run ace2Oligos.perl on that ace file and direct the oligo file to be in the parent directory of edit_dir, phd_dir, and chromat_dir for that project. Thus there will be one oligos file for each project. Your script will run ace2Oligos.perl once for each project.

Then your script would, for each project, look in the oligos file for new oligos, and aggregate the unordered oligos into a central file, which it would email to the oligo company. If it finds any new oligos in an oligo file, it draws a line at the bottom:

```
------------------------------
```

which indicates that all oligos have been ordered. When this script looks at this file the next night, it uses this line to determine whether any additional oligos have been requested since the previous order. (The idea of this line came from St Louis.) Thus the oligos file tells you which oligos have been ordered and which have not yet been ordered.

183) USER-DEFINED CONSENSUS POSITIONS

Suppose instead of labeling the consensus 1, 2, 3, 4, ..., you want, for example, to number it: 100,000,001, 100,000,002, 100,000,003, 100,000,004, etc. (e.g., in chromosome positions). You can do this. Note that all bases in the consensus (except pads) will be numbered--you cannot, for example, only number exon bases and not number intron bases (pity).

To start numbering the consensus at a number different from 1, add a "startNumberingConsensus" tag to either the consensus or a read in that contig. The tag will look like this (this is a consensus tag in the ace file):

```
CT{
hs18-25105605_HSap-Contig startNumberingConsensus consed 1 1 041123:152840
25105605
}
```

This says that the consensus will be numbered starting at 25,105,605

You cannot add such a tag by using Consed--you must have a program add it to the ace file (or a phd file of one of the reads in the contig).

184) CUSTOM NAVIGATION

In the Main Window, there is also a Navigate menu. Pull it down and release on the Custom Navigation menu item. A box will pop up saying 'Select custom navigation file:'

There will be a file:
custom_navigation.nav
Double click on it.

You will see the now-familiar custom navigation box. Click 'Next' repeatedly until you get to the end of the list.

Consed doesn't write such a file--it just reads it. This feature allows you the ability to write your own programs that select locations that you want your finishers to examine. Your program writes a file, the user reads that file into Consed in this manner, and you can go to each of the locations.

The format of the file is as follows:

There is a title line that looks like this:

```
TITLE: low quality base in discrepant region
```

and then there are blocks that look like this.

```
BEGIN_REGION
TYPE: READ
CONTIG: hs2-105068850_HSap-Contig
READ: E02_hs2-105068850_PTro_040520.f
UNPADDED_CONS_POS: 295 299
COMMENT: left 2
END_REGION
```

The block above refers to a position on read
E02_hs2-105068850_PTro_040520.f in contig hs2-105068850_HSap-Contig at consensus positions 295-299.

There is another kind of block:

```
BEGIN_REGION
TYPE: CONSENSUS
CONTIG: hs21-15002178_HSap-Contig
UNPADDED_CONS_POS: 1774 1784
COMMENT: another comment
END_REGION
```

which refers to a position on the consensus. Notice that it is missing the "READ:" line and the TYPE: line is different. When someone is navigating, the blinking cursor will be put onto the consensus (with the second kind of block) rather than the blinking cursor on the read (with the first kind of block).

You might want to specify the consensus positions in terms of some user-defined positions (the first position of the consensus is not 1 but rather is some other number). For example, you might want to use chromosome positions, rather than the position within the contig. You can let Consed know that the UNPADDED_CONS_POS numbers are user-defined positions by putting the words "user-defined positions" somewhere in the TITLE line. Of course, you must also have a startNumberingConsensus tag on the consensus or a read indicating the user-defined position of the left-end of the contig.

185) DEFINING KEYS (HOTKEYS) TO CALL EXTERNAL PROGRAMS AND/OR APPLY TAGS AND/OR INTEGRATE CONSED WITH EXTERNAL DATABASES

[CUSTOM KEYS, USER-DEFINED KEYS]

You can define keys (such as Control-N) to apply a particular tag to a single base, saving you the several steps in applying tags: swiping and selecting a tag type (as shown under "TAGS" above). However, it is even more powerful than that. You can also define an external program to run when you type this key. That external program can be your own, and it could be, for example, a program that puts information into an external database.

The first thing you need to set up a custom hotkey is a .consedrc file which goes in edit_dir of the project you're working on (see above CONSED CUSTOMIZATION for other possible locations).

Put the following in that file:

```
consed.userDefinedKeys: 14 15
! make a space-separated list of the decimal ASCII values of the keys
! 14 means control-N, 15 means control-O

consed.programsForUserDefinedKeys: /bin/echo /bin/echo
! a space-separated list of the full pathnames of the commands to run

consed.argumentsToPassToUserDefinedPrograms: argument_for_first_key argument_for_se
cond_key
! a space-separated list of the arguments to pass to each user-defined programs

consed.tagsToApplyWithUserDefinedKeys: none polymorphismConfirmed
! a space-separate list of the tag types to apply when the user
! presses a user-defined key. If a key is to have no associated tag,
! then enter "none" for that key.
```

This makes control-N and control-O ("oh"--not zero) call "/bin/echo" by default. In either the aligned reads window or the trace window, click the cursor on a base and try these keys (e.g., holding down the control key and typing 'o'). Watch in the xterm where you started Consed for output like this:

```
argument_for_first_key djs74-561.s1 97 Contig1 2534 2581 a 51 /kw3/gordon/consed_demo/standard/edit_dir/standard.fasta.screen
argument_for_second_key djs74-2679.s1 78 Contig1 2527 2574 c 39 /kw3/gordon/consed_demo/standard/edit_dir/standard.fasta.scre
```

```
djs74_561.s1 the read the user was viewing (or "consensus" if the
    cursor is on the consensus)
97 the base position in the direction of sequencing (or -1 if the
    cursor is on the consensus)
Contig1 the contig
2534 the unpadded consensus position
```

2581 is the padded (counts *'s) consensus position
'a' is the base
51 is the quality of the base
/kw3/gordon/consed_demo/standard/edit_dir/standard.fasta.screen.ace.1 is the ace file
tr.window means it was called by the user pushing the key in the trace
window--not the aligned reads window.

It's the same as if you had run the
program from the shell, with command-line arguments, like this:

```
bash%: /bin/echo argument_for_first_key djs74-561.s1 97 Contig1 2534 2581 a 51 /kw3/gordon/consed_demo/standard/edit_dir/sta
```

You will also see that control-O will automatically add a
polymorphismConfirmed tag, but control-N will not add any tag. That
is because of consed.tagsToApplyWithUserDefinedKeys (see above).

Several groups that are doing polymorphism detection have expressed
interest in this feature because it enables them to have Consed
directly write into an external database (e.g., Oracle or Sybase) by
calling a program that then writes to the database.

You can use these hotkeys from within the trace window or the aligned
reads window.
You don't have to use only ctrl-N/ctrl-O... for instance 1 is
control-A, 2 is control-B, 3 is control-C, 4 is control-D, etc.

If you want to pass this information to a database, you will need to know
how to talk to your database, and either choose your hotkey to do it
directly for you, or call another program that takes the parameters
above and massages them into the format your database needs.

control-A, control-E, and control-T already mean something in the
aligned reads window, so those keys cannot be defined to be anything
else. Typically control-C, control-S, and control-Q already mean
something to the operating system so you can't use those, either.

186) READ PREFIXES

You can create a file called readPrefixes.txt in edit_dir. This file
contains a list of reads and prefixes for those reads. In the Aligned
Reads Window, the Consed user will see those read prefixes in a column
before the read names. This can be a very helpful feature for
finishers. For example, these read prefixes can indicate to the
finishers which templates are available to use for making finishing
reads.

The format of the file is:

```
(readname) (read prefix) (color for read prefix)
```

The read prefix and color for read prefix are optional. If you
leave them out, you get '*' for the read prefix in blue.

The consed parameters involving this feature are:

```
consed.defaultReadPrefix: *
consed.readPrefixesFile: readPrefixes.txt
consed.maxCharsDisplayedForReadPrefix: 1
```

but you probably won't need to change them.

187) USING FILES CREATED ON WINDOWS OR WINDOWS NT.

Don't. (E.g., phd files generated by a Beckman CEQ-2000.) These
files initially had <CR><LF> at end of line instead of <LF>. CONSED
chokes every time it tries to read something from these phd files.
If you must use these files, you must first convert them to UNIX
format, which means stripping out the CR's and just having \n (decimal 10)
separate lines.

188) CREATING YOUR OWN ACE FILES (INSTEAD OF ACE FILES CREATED BY
PHRAP)

Some people have tried creating their own ace files, try Consed on it,
and when Consed starts up ok, they don't understand when later some
feature in Consed doesn't work. This is because Consed does not check
everything about an ace file when it starts up. If you are going to
write software to create ace files, here is a partial list of Consed
features you should check before you think your ace files are fine for
Consed:

assembly view
restriction digest
read all traces
complement contig and then read all traces
add new reads

If all of these work properly, then your ace files are probably ok.

MONITORS AND MICE FOR CONSED

If your monitor is part of a Unix computer (a Sun, an HP, a DEC, an
SGI, or a Linux box) or is an Xterminal, then you will have absolutely
no problems.

You must have 3 button mouse or 3 button emulation. 3 Button
emulation is tricky since Consed uses all 3 buttons of the mouse and
it also uses Control-Middle-Mouse-button, Shift-Middle-Mouse-Button
and Control-Right-Mouse-Button. So if you are going to try to just
use a 2 button mouse (or, God-forbid, a 1 button mouse), you should
make sure that you can emulate each of those. Often, if you push the
left and right mouse buttons at the same time, your X server will

interpret that to be the middle mouse button. But you must consult your X emulator or X server to know what it will do--that is out of Consed's control.

If your monitor is a PC running Windows or NT, then you must have an X emulator installed and running. X emulators include: Exceed, XWin32, Reflection X, and OpenNT. Any of these will work if configured correctly (and the 'correctly' is the key). I encourage you to use single window mode and then use a Unix window manager such as CDE, fvwm, or mwm.

If your monitor is a MAC with macosx running, see NOTE TO MACOSX USERS (above).

If your monitor is a MAC (pre-Macosx), then you must also have an X emulator, such as Exodus or MACX installed and running. You *must* use this emulator in single window mode, and then use a Unix window manager such as CDE, fvwm, or mwm. (If you don't use single window mode, Consed might crash in some circumstances.)

AUTOFINISH AND PRIMER-PICKING PARAMETERS AND OTHER PARAMETERS

Some of the parameters below are used by Autofinish, some by Consed's primer picker, and some by both.

You should use the default values of these parameters unless you have a particular reason for changing them. The defaults have been chosen very carefully based on theory and experimentation and are the ones being used at the major genome centers.

You can set these via the .consedrc file--see CONSED CUSTOMIZATION (above)

In addition, for a particular Consed session, you interactively change many of these in the following manner: On the main window, point to 'Options', hold down the left mouse button and release on 'Primer Picking Preferences.' You can modify the parameter of interest and then click on 'Apply and Dismiss'. The new value of the parameter will be in affect only until you restart Consed.

For the most current list, in the Consed Main Window, point to 'Info', hold down the left button, and release on 'Show Current Consed Parameters'.

```
! In the following, I have annotated the parameters with the following
! symbols:
!
! (YES)   freely customize to your own site
! (OK)   don't change unless you have a specific need and know what you
!          are doing
! (NO)   don't change this!
!
!
!
! parameters in the (YES) category:
!
consed.autoFinishMinNumberOfErrorsFixedByAnExp:  0.02
double
! if an experiment solves fewer errors than this, it isn't worth doing
! so won't be chosen.  This parameter controls when Autofinish stops
! choosing experiments.
! (YES)

consed.autoFinishRedundancy:  2.0
double
! This number should be between 1.0 and 2.0 If you want more reads
! for each area, increase the number towards 2.0  If you want fewer
! reads per area, decrease it towards 1.0.  This only affects
! universal primer reads--not custom primer reads.
!
! (YES)

consed.autoFinishAverageInsertSize:  1500
int
! If a template has a forward but no reverse, when deciding whether to
! allow this template for a particular primer or reverse, we need to
! make an assumption of where is the end of the template.  If we have
! do not have enough forward/reverse pairs to determine the mean, then
! this parameter is used.
! (YES)

consed.primersMaxInsertSizeOfASubclone:  3000
int
// for checking for false-annealing
! check +/- this distance from the primer for false-annealing
! and check at most this distance for templates for a primer.
! Thus if you have more than one library, make this the max of
! all libraries.
! (YES)

consed.primersMaxMeltingTemp:  60
int
! (YES)

consed.primersMaxMeltingTempForPCR:  58
int
! Note:  the difference between consed.primersMaxMeltingTempForPCR and
! consed.primersMinMeltingTempForPCR must be less than or equal to
!  consed.primersMaxMeltingTempDifferenceForPCR
! Otherwise, autofinish may take forever to pick pcr primers.
! (YES)

consed.primersPickTemplatesForPrimers:  true
bool
! when picking primers for subclone templates, pick templates also.
! If there is no suitable template for a primer, do not pick the
! primer.  If you like to pick your own templates, you might want to
! turn this off for a little improvement in speed.
! This has no effect on Autofinish--just on interactive primer picking
! in Consed.
! (YES)

consed.primersSubcloneFullPathnameOfFileOfSequencesForScreening:  $CONSED_HOME/lib/screenLibs/primerSubcloneScreen.seq
RWCString
! vector sequence file if choosing subclone (e.g., M13, plasmid)
! templates
! (YES)

consed.primersCloneFullPathnameOfFileOfSequencesForScreening:  $CONSED_HOME/lib/screenLibs/primerCloneScreen.seq
RWCString
! vector sequence file if choosing clone (e.g., cosmid, BAC) template
! (YES)

consed.primersMinMeltingTemp:  55
int
! (YES)

consed.primersMinMeltingTempForPCR:  55
int
! (YES)

consed.searchFunctionsUseUnalignedEndsOfReads:  false
bool
! when navigating by
!  searchForSingleSubcloneRegions  and  searchForSingleStrandedRegions,
! and the read below has both aligned and unaligned portions, which
! bases of the read are considered to cover the region:
!     uuuuuuuAAAAAAAAAAAAAAAAAAAAAAAAAAuuuuuuuu
!    <--------- if "true" ------------------>
!            <-----if "false"-------->
! where u means an unaligned base and A means an aligned base
! (YES)

consed.searchFunctionsUseLowQualityEndsOfReads:  true
bool
! when navigating by
!  searchForSingleSubcloneRegions  and  searchForSingleStrandedRegions,
! and the read below has both low quality and high quality portions,
! which portions of the read are considered to cover the region:
!     lllllllAAAAAAAAAAAAAAAAAAAAAAAAAAllllllll
!    <--------- if "true" ------------------>
!            <-----if "false"-------->
! where l means a low quality base and A means a high quality base
! (YES)

consed.inexactSearchForStringMaxPerCentMismatch:  5
int
! when using the inexact search for string, allow up to this
! % mismatch:  the sum of the insertion, deletion, and substitution
! differences divided by the length of the query string
! (YES)

consed.onlyAllowOneReadWriteConsedAtATime:  false
bool
! if there is another read-write consed (or Autofinish) process running in the
! same directory, and this consed (or Autofinish) is not read-only,
! then terminate with an error message
! (YES)

consed.autoFinishAllowHighQualityDiscrepanciesInTemplateIfConsistentForwardReversePair:  true
bool
!   otherwise, a single serious hqd will cause the template to be rejected.
! (YES)


consed.printWindowCommand:  /usr/bin/X11/xwd  |  /usr/bin/X11/xpr  |  /bin/lp  -dlevulose
RWCString
! system command to print out a Consed Window
! (YES)

consed.fileOfTagTypes:
FileName
! pathname of a file with the following format:
! (tag name) (color for displaying) (consensus or read or both) (yes/no)
! where "consensus" or "read" or "both" indicates whether the tag
! is available for the user to add to the consensus, to reads, or to
! both, and "yes" or "no" indicates whether the tag can be created
! in Consed by swiping, or whether it only can be created by an
! external program and displayed by Consed.
! (YES)

consed.assemblyViewShowConsistentFwdRevPairs:  false
bool
! too many squares!   See assemblyViewShowConsistentFwdRevPairDepth
! (YES)

consed.assemblyViewShowConsistentFwdRevPairDepth:  true
bool
! This actually shows more information than
! assemblyViewShowConsistentFwdRevPairs      and is much easier to read
! (YES)

consed.assemblyViewShowConsistentFwdRevPairsBetweenDifferentScaffolds:  true
bool
! Lone links from the end of one contig to the end of another, but not
! confirmed by another in order to make the contigs joined into a scaffold.
! (YES)

consed.assemblyViewShowLegsOnSquaresForConsistentFwdRevPairs:  false
bool
! This is even more cluttered than assemblyViewShowConsistentFwdRevPairs
! (YES)

consed.assemblyViewShowGapSpanningFwdRevPairs:  true
bool
! This shows gap-spanning fwd/rev pairs that caused the contigs to
! be joined into a scaffold.
! (YES)
```

```
consed.assemblyViewShowWhichInconsistentFwdRevPairs:  filtered
RWCString
! choices are: filtered, none, all
! "filtered" means that an inconsistent fwd/rev pair is only shown
! if it is confirmed by another inconsistent fwd/rev pair
! If all, full of red lines.  If filtered, then only red lines that are
! confirmed by other red lines are shown.
! (YES)

consed.assemblyViewShowReadDepth:  true
bool
! If true, read depth is shown in assemblyView
! (YES)

consed.assemblyViewShowRestrictionDigestCutSites:  true
bool
! If true, and you open a Digest Window in Consed and you open
! the Assembly View window in Consed, the restriction digest cut
! sites will be shown in Assembly View (in addition to showing them in
! the Digest Window)
! (YES)

consed.assemblyViewFilterSequenceMatchesBySize:  false
bool
! only show sequence matches if they fall between
!  consed.assemblyViewSequenceMatchesMinSize  and
!  consed.assemblyViewSequenceMatchesMaxSize
! (YES)

consed.assemblyViewSequenceMatchesMinSize:  100
int
!  if  consed.assemblyViewFilterSequenceMatchesBySize  is  true,
! then only show sequence matches that are larger than this
! (YES)

consed.assemblyViewSequenceMatchesMaxSize:  10000
int
!  if  consed.assemblyViewFilterSequenceMatchesBySize  is  true,
! then only show sequence matches that are smaller than this
! (YES)

consed.assemblyViewAutomaticallyStartWithConsed:  false
bool
! when consed starts, start assembly view.   This only works if you
! specify the ace file on the command line.
! (YES)

consed.assemblyViewDisplayTheseTagTypesOnTheseLines:  edit  0  matchElsewhereHighQual  1  matchElsewhereLowQual  2
RWCString
! space-separated list of form:
! (tagtype) (line number) (tagtype) (line number)
! where line number is where in Assembly View the tag will be displayed
! (YES)

consed.assemblyViewShowTags:  true
bool
! If true, and some tag types are selected, these tags
! will be shown in assemblyView.   If false, no tags
! will be shown in assemblyView.
! (YES)

consed.autoEditRecalculateHighQualitySegmentsOfReads:  false
bool
! If true, will recalculate the high quality segments of the reads
! (YES)

consed.autoEditConvertCloneEndBasesToXs:  true
bool
! If true, will convert to X's bases of all reads that protrude beyond a
! cloneEnd tag.
! (YES)

consed.autoEditTellPhrapNotToOverlapMultiplyDiscrepantReads:  true
bool
! This will find all locations where there are multiple identical
! discrepancies with the consensus (and some other conditions) and try
! to make most of the reads quality 99 at that location so that phrap,
! next time it is run, will not overlap those reads.   This will fix
!  many misassemblies.
! (YES)

consed.autoEditTagEditableLowConsensusQualityRegions:  true
bool
! This will find regions that are low quality, but that a human
!  finisher could easily determine the correct base and thus
! money could be saved by not having Autofinish suggest additional
! reads overlapping the region
! (YES)

consed.showAllTracesJustShowGoodTraces:  true
bool
! Just show traces where there is a base at the cursor and
! there is trace signal at the cursor and where
! there is no "dataNeeded" tag at the cursor as specified by
!  consed.showAllTracesDoNotShowTraceIfTheseTagsPresent
! (YES)

consed.addAlignedSequenceQualityOfBases:  40
int
! when running consed -addAlignedSequence, what quality should the
! bases be?
! (YES)

consed.makeLightBackgroundInAlignedReadsWindowAndTracesWindow:  false
bool
! for printing screens, saves toner
! (YES)

!
```

```
!
! parameters in the (OK) category:
!
!
!

consed.autoReportCompareTopAndBottomStrands:  false
bool
! (OK)

consed.autoReportPrintLengthsOfAlignedSegmentsOfReads:  false
bool
! (OK)

consed.autoReportPrintLengthsOfUnalignedHighQualitySegmentsOfReads:  false
bool
! (OK)

consed.autoReportPrintIfReadsAreCorrectlyAligned:  false
bool
! make sure that .f reads are top strand and .r reads are bottom
! strand (Note:  this only is true in some projects.)
! (OK)

consed.autoReportPrintScaffolds:  false
bool
! (OK)

consed.autoReportCalculateErrorProbabilitiesByComparingPTroPPan:  false
bool
! (OK)

consed.autoReportPrintAgreeDisagreeBetweenPairsOfSpecies:  false
bool
! (OK)

consed.autoReportPrintAgreeDisagreeBetweenPairsOfSpecies2:  false
bool
! differs from above in that one or the other of the species bases
! must be at least quality 45
! (OK)

consed.autoReportQualityWindowLow:  10
int
! (OK)

consed.autoReportQualityWindowHigh:  15
int
! (OK)

consed.autoReportPrintNumberOfIsolatedPadsForEachSpecies:  false
bool
! (OK)

consed.autoReportPrintNumberOfIsolatedPads:  false
bool
! (OK)

consed.autoReportIsolatedPadsOfReadsWithThisPattern:  PTro
RWCString
! (OK)

consed.autoReportMinNumberOfPerfectlyAlignedBasesBeforeDiscrepancy:  5
int
! (OK)

consed.autoReportPrintMinimumQualityHistogram:  false
bool
! (OK)

consed.autoReportPrintDiscrepantRegions:  false
bool
! (OK)

consed.autoReportPrintBasesInDiscrepantRegions:  false
bool
! (OK)

consed.autoReportPrintDiscrepantRegionsButIgnoreReadsContainingThis:
RWCString
! (OK)

consed.autoReportBackboneReadHasThisStringInIt:  HSap
RWCString
! (OK)

consed.numberUnpaddedConsensusAtUserDefined:  true
bool
! allow use to put a tag on the consensus to specify the number to
! start numbering the consensus
! (OK)

consed.autoReportPrintDiscrepantRegionsButOnlyIfAboveQualityThreshold:  false
bool
! if true,
!  uses  consed.qualityThresholdForFindingHighQualityDiscrepancies
! (OK)

consed.autoReportPrintSpeciesAlignment:  false
bool
! (OK)

consed.autoReportSpecies: PPan  PTro  GGor  PPyg  MMul
RWCString
! These are just used for autoReportPrintSpeciesAlignment
! (OK)

consed.autoReportPrintReadAlignment:  false
bool
! This differs from consed.autoReportPrintSpeciesAlignment in that
! reads that are for the same species are not combined.
! (OK)

consed.autoReportPrintTheseReads:  readsToPrint.txt
FileName
! (OK)

consed.autoReportPrintReadPositions:  false
bool
! (OK)

consed.autoReportPrintChosenReadName:  false
bool
! (OK)

consed.autoReportNumbersOfCharactersOfChosenReadNameToBePrinted:  1
int
! if the read name is larger than this, will print this number
! of characters at the end of the name.  If the read name is shorter,
! will just print the read name.
! (OK)

consed.autoReportDumpQualities:  false
bool
! (OK)

consed.autoReportPrefix:  1
RWCString
! normally this will be the chromosome #
! (OK)

consed.autoReportMaxSizeOfDiscrepantRegion:  3
int
! used for printing bases of a discrepant region
! (OK)


consed.showAllTracesDoNotShowTraceIfTheseTagsPresent:  dataNeeded
RWCString
!  See  consed.showAllTracesJustShowGoodTraces
! (OK)

consed.nameOfFakeJoiningReadsIncludesAceFileName:  false
bool
! This is useful if the user is going to combine the reads
! from a number of different ace files together.
! (OK)

consed.whenUserScrollsOffWindowMillisecondsBetweenScrolling:  250
int
! (OK)

consed.whenUserScrollsOffWindowBasesToScrollEachTime:  15
int
! (OK)

consed.compareContigsUseBandedRatherThanFullSmithWaterman:  true
bool
! (OK)

consed.compareContigsBandSize:  50
int
! band size of banded Smith Waterman
! (OK)

consed.assemblyViewShowFwdRevPairDepthsInRedIfOnlyThisMany:  1
int
! (OK)

consed.assemblyViewShowSequenceMatches:  true
bool
! When false, do not show any sequence matches (repeats)
! at all in Assembly View.
! Some people like to start out this way since displaying sequence
! matches slows down scrolling.
! (OK)

consed.assemblyViewOKToShowSequenceMatchesBetweenContigs:  true
bool
! (OK)

consed.assemblyViewOKToShowSequenceMatchesWithinContigs:  true
bool
! (OK)

consed.assemblyViewOKToShowDirectSequenceMatches:  true
bool
! This means in which neither copy must be complemented with respect
! to the way it is in the scaffold as created by Consed.
! (OK)

consed.assemblyViewOKToShowInvertedSequenceMatches:  true
bool
! This means that exactly one copy must be complemented with respect
! to the way it is in the scaffold as created by Consed.
! (OK)

consed.assemblyViewOnlyShowSequenceMatchesToAParticularRegion:  false
bool
!  You  must  set  consed.assemblyViewOnlyShowSequencematchesToThisContig
!  consed.assemblyViewOnlyShowSequenceMatchesToThisRegionLeft
!  consed.assemblyViewOnlyShowSequenceMatchesToThisRegionRight
! (OK)

consed.assemblyViewOnlyShowSequenceMatchesToThisContig:
RWCString
! You must make
!  consed.assemblyViewOnlyShowSequenceMatchesToAParticularRegion:  true
! (OK)
```

```
consed.assemblyViewOnlyShowSequenceMatchesToThisRegionLeft:  0
int
!  consed.assemblyViewOnlyShowSequenceMatchesToAParticularRegion:  true
! (OK)

consed.assemblyViewOnlyShowSequenceMatchesToThisRegionRight:  0
int
!  consed.assemblyViewOnlyShowSequenceMatchesToAParticularRegion:  true
! (OK)

consed.defaultReadPrefix:  *
RWCString
! This is used as the character to prefix reads with when the
! read is in consed.readPrefixesFile and the prefix is not specified.
! (OK)

consed.readPrefixesFile:  readPrefixes.txt
FileName
! This file should contain a list of reads that you would want to
! have prefixes in the Aligned Reads Window.  Each line should
! have the following format:
! (read name) (prefix) (color)
! The prefix and color are optional.  You can have a line like this:
! (read name) (prefix)
! or this:
! (read name)
! but not this:
! (read name) (color)
! If the color is not specified, the color will default to
!  consed.colorReadPrefixes:  blue
! If the prefix is not specified, it will default to
! (OK)

consed.maxCharsDisplayedForReadPrefix:  1
int
! It is still ok to have long read prefixes in the file
! consed.readPrefixesFile but only this many characters
! will be displayed in the Aligned Reads window
! (OK)

consed.autoFinishDoNotDoPCRIfThisManyAvailableGapSpanningTemplates:  2
int
! (OK)

consed.autoFinishDoNotDoUnorientedPCRIfThisManyOrMoreUnorientedPCRReactions:  6
int
! "unoriented" pcr reactions means cases in which autofinish is suggesting
! a pcr reaction to span a gap, but it doesn't know whether the 2 contig ends
! really go together since there are not enough (or no)templates that span
! that gap
! (OK)

consed.autoFinishDoNotDoOrientedPCRIfGapSizeLargerThanThis:  10000
int
! Gap size can be specified in user-defined contigEndPair tags in a
! gap_size: field
! If the gap size is greater than this number, do not do PCR.
! (OK)

consed.autoFinishDoNotDoPCRIfEndIsExtendedByReads:  false
bool
! If this is true, and autofinish was able to walk off the end of a
! contig, do not do PCR with that end of the contig.
!
! (OK)

consed.autoFinishMaxAcceptableErrorsPerMegabase:  0
int
! target error rate.  This parameter used to be the one that stopped
! Autofinish from calling more reads.  However, consider a BAC that is
! nearly perfect except for one region with 3 quality 10 bases in a
! row.  In this case the global errors per megabase is very
! low--perhaps lower than 1 error per megabase.  Despite this, most
! labs would like to do one more read to fix this problem.  Thus we
! set this parameter to zero (to disable it) so Autofinish will use
! the parameter consed.autoFinishMinNumberOfErrorsFixedByAnExp to stop
! calling more reads--it is a local error rate.
! (OK)

consed.autoFinishIfNotEnoughFwdRevPairsUseThisPerCentOfInsertSize:  90
int
! If a template has a forward but no reverse, when deciding whether to
! allow this template for a particular primer, we need to make an assumption
! of where is the end of the template.  If the template comes from a library
! with insert size 1500, it would be reasonable to assume that the end of
! template will be 1500 bases from the forward read.  But if this template
! has an insert that is shorter than average, the walk may walk into vector.
! To be conservative, we may want to assume that the insert is somewhat
! shorter than average.  By default, we assume that it is 90% as large as
! the average. This parameter gives that percentage.  This parameter
! is used both by Consed and Autofinish.
! (OK)

consed.primersNumberOfBasesToBackUpToStartLooking:  50
int
! e.g., if this is 50 and you want a read at position 1000, primers
! will be searched before base 950 but not in the region 950 to 1000
! This has no effect on Autofinish--just on interactively picking primers.
! (OK)

consed.primersMakePCRPrimersThisManyBasesBackFromEndOfHighQualitySegment:  100
int
! When a PCR product is made, you want it to overlap by this many bases
! the high quality part of the existing consensus.  Thus choose PCR
! primers this many bases back (or more)
! (OK)

consed.primersOKToChoosePrimersInSingleSubcloneRegion:  true
bool
! (OK)
```

```
consed.primersOKToChoosePrimersWhereHighQualityDiscrepancies:  false
bool
! (OK)

consed.primersOKToChoosePrimersWhereUnalignedHighQualityRegion:  false
bool
! (OK)

consed.autoFinishCallReversesToFlankGaps:  true
bool
! if there is a forward-reverse pair flanking a gap, print it out
! if there is not, suggest reverses to flank the gap
! (OK)

consed.autoFinishAllowWholeCloneReads:  false
bool
! ok to call reads whose template for sequencing reaction is the
! entire clone (BAC or cosmid)
! (OK)

consed.autoFinishAllowCustomPrimerSubcloneReads:  true
bool
! ok to call reads with custom primers and subclone template
! (OK)

consed.autoFinishAllowResequencingReads:  true
bool
! This is just universal primer reads to be resequenced using
! dye terminator chemistry or special chemistry.  (It does not
! mean resequencing a custom primer read.)
! (OK)

consed.autoFinishAllowResequencingReadsOnlyForRunsAndStops:  false
bool
! This parameter only has any effect when
! consed.autoFinishAllowResequencingReads is set to true.  In that
! case no resequencing reads will be suggested, unless it is to cross
! a run or stop and special chemistry is suggested.
! (OK)

consed.autoFinishAllowDeNovoUniversalPrimerSubcloneReads:  true
bool
! Allows calling reverse when there is just a forward.
! Allows calling a forward when there is just a reverse.
! (OK)

consed.autoFinishAllowMinilibraries:  false
bool
! Allows calling minilibraries (shatter libraries or transposon
! libraries) of subclone templates for closing gaps
! (OK)

consed.autoFinishAllowPCR:  true
bool
! Allows calling PCR for closing gaps, but only as a last resort
! (OK)

consed.autoFinishAllowUnorientedPCRReactions:  true
bool
! Allows calling PCR amongst contig-ends that have insufficient
! fwd/rev pair linkage to any other contig-end.  Thus it suggests
! pcr amongst all such contig-ends.
! To allow this type of pcr, you must also make:
!  consed.autoFinishAllowPCRForUnorientedContigEnds:  true
! See also:
!  consed.autoFinishDoNotDoUnorientedPCRIfThisManyOrMoreUnorientedPCRReactions:
! which gives you finer control over unoriented pcr.
! (OK)

consed.autoFinishAllowResequencingAUniversalPrimerAutofinishRead:  false
bool
! if Autofinish suggests a de novo universal primer read,
! do not allow Autofinish to suggest a resequence of this read
! (OK)


consed.autoFinishAlwaysCloseGapsUsingMinilibraries:  false
bool
! "Minilibraries" includes transposing a subclone template or
! making a shatter library from a subclone template
! (OK)

consed.autoFinishMaximumFinishingReadLength:  2000
int
! Change this only if your finishing reads are typically shorter
! than your shotgun reads.  Otherwise, leave it unrealistically long,
! and Autofinish will set its model read based on your existing
! shotgun reads.
! (OK)

consed.autoFinishSuggestMinilibraryIfGapThisManyBasesOrLarger:  800
int
! (OK)

consed.autoFinishSuggestSpecialChemistryForRunsAndStops:  true
bool
! Suggest special chemistry such as dGTP for reads that cross
! mononucleotide or dinucleotide repeats that cause reads to fail or
! stops (structure) that cause reads to fail and thus dye terminator
! reads won't work.
! (OK)


consed.autoFinishSuggestThisManyMinilibrariesPerGap:  2
int
! (OK)

consed.primersWindowSizeInLooking:  450
int
```

```
! e.g., if this is 300, with example above, primers will be searched
! from base 650 to 950.  This has no effect on Autofinish--it is just
! used for interactive primer picking in Consed.
! (OK)

consed.primersAssumeTemplatesAreDoubleStrandedUnlessSpecified:  false
bool
! you can put the template type in the phd file in a WR template item
! consed will have a list of these and know which are single and
! double stranded
! (OK)

consed.alignedReadsWindowInitialCharsWide:  60
int
! initial width of the aligned reads window including the read name and
! the bases
! (OK)

consed.alignedReadsWindowInitialCharsHigh:  20
int
! initial height of the aligned reads window area where the consensus
! and reads are
! (OK)

consed.alignedReadsWindowMaxCharsForReadNames:  20
int
! how many columns are reserved for read names
! (OK)

consed.alignedReadsWindowAutomaticallyExpandRoomForReadNames:  true
bool
! If true, expand and contract space for read names, but don't
!  contract  less  than  consed.alignedReadsWindowMaxCharsForReadNames.
! If false, then always use
!  consed.alignedReadsWindowMaxCharsForReadNames
! for space reserved for read names.
! (OK)

consed.autoFinishAllowResequencingReadsToExtendContigs:  false
bool
! if false, a resequencing read is not called to extend a contig--only
! custom primer reads and de novo universal primer reads are called
! for this purpose.
! (OK)

consed.autoFinishCallHowManyReversesToFlankGaps:  2
int
! This has two purposes: 1) it specifies how many forward/reverse
! pairs should be present for Consed/Autofinish to be certain of the
! order/ orientation of two contigs.   If there are this many fwd/rev
! pairs flanking a gap, Autofinish will print out the contig ends that
! flank the gap.   2) If consed.autoFinishCallReversesToFlankGaps is
! set to true, and there are less than this many fwd/rev pairs
! flanking a gap, Autofinish will suggest additional reverses until
! there are this many.
! (OK)

consed.autoFinishCloseGaps:  true
bool
! this allows you to turn off choosing reads to close gaps
! (OK)

consed.autoFinishContinueEvenThoughReadInfoDoesNotMakeSense:  false
bool
! this allows you to override the checks that autofinish makes on the
! read info, such as checking there are not more than 5 or so reads
! from the same subclone template
! (OK)

consed.autoFinishCostOfResequencingUniversalPrimerSubcloneReaction:  20.0
double
! compares universal primer subclone reaction, custom primer subclone
! reaction, and custom primer clone reaction to decide which to favor
! (OK)

consed.autoFinishCostOfCustomPrimerSubcloneReaction:  60.0
double
! see above
! (OK)

consed.autoFinishCostOfCustomPrimerCloneReaction:  80.0
double
! see above
! (OK)

consed.autoFinishCostOfDeNovoUniversalPrimerSubcloneReaction:  60.0
double
! cost of reverse where there is only a forward or cost of forward
! when there is only a reverse
! (OK)

consed.autoFinishCostOfMinilibrary:  500.0
double
! cost of making a minilibrary (transposon library or shatter library)
! from a subclone template
! (OK)

consed.autoFinishCoverSingleSubcloneRegions:  true
bool
! this allows you to turn off choosing reads to cover single subclone regions
! (OK)

consed.autoFinishCoverLowConsensusQualityRegions:  true
bool
! this allows you to turn off choosing reads to cover low consensus
! quality regions
! (OK)

consed.autoFinishDebugUniversalPrimerReadsFile:  gordon_debug.txt
FileName
```

```
! for debugging Autofinish
! put a file with this name in the same directory as the ace file
! format:
! fcalld09 fwd
! fgj74f01 rev
! (template name) (fwd or rev)
! (OK)

consed.autoFinishDebugCustomPrimerReadsFile:  debug_custom.txt
FileName
! for debugging Autofinish
! put a file with this name in the same directory as the ace file
! format:
! cgggacctgg
! (primer in 5' to 3' orientation)
! (OK)

consed.autoFinishDoNotAllowSubcloneCustomPrimerReadsCloserThanThisManyBases:  200
int
!  see  consed.autoFinishDoNotAllowSubcloneCustomPrimerReadsCloseTogether
! (OK)

consed.autoFinishDoNotAllowWholeCloneCustomPrimerReadsCloserThanThisManyBases:  300
int
!  see  consed.autoFinishDoNotAllowWholeCloneCustomPrimerReadsCloseTogether
! (OK)

consed.autoFinishDoNotFinishWhereTheseTagsAre:  doNotFinish  editable
RWCString
! list of tag types separated by spaces.  E.g.,
! doNotFinish repeat
! tells autofinish that you are not interested in finishing in this region
! (OK)

consed.autoFinishDoNotExtendContigsWhereTheseTagsAre:  doNotFinish
RWCString
! list of tag types separated by spaces.  E.g.,
! doNotFinish repeat
! tells autofinish that you do not want to extend the contig near this
! tag.  If you do not want this feature, just leave the list empty.
! (OK)

consed.autoFinishDoNotExtendContigsIfTagsAreThisCloseToContigEnd:  50
int
! Uses the list from consed.autoFinishDoNotExtendContigsWhereTheseTagsAre
! and checks if any of these tags are within this many bases of the end of
! the contig.  If they are, does not extend the contig.
! (OK)

consed.dumpContigOrderAndOrientationInfoToThisFile:
FileName
! In the case of Consed (not autofinish or autoPCRAmplify), send the
! output to this file rather than stderr.  If this name is blank,
! continue (in case of consed), to send output to stderr.
! (OK)

consed.autoFinishDumpTemplates:  false
bool
! for debugging, this allows you to dump all information about the
! templates--insert locations
! (OK)

consed.autoFinishExcludeContigIfOnlyThisManyReadsOrLess:  10
int
! (OK)

consed.autoFinishExcludeContigIfDepthOfCoverageGreaterThanThis:  50.0
double
! To exclude contigs that are probably E. coli contamination
! "depth of coverage" is defined here to mean the sum of the read
! lengths (including low quality ends) divided by the contig length.
! (OK)

consed.autoFinishExcludeContigIfThisManyBasesOrLess:  1000
int
! consed.autoFinishExcludeContigIfTooShort must be set to true for
! this to have any effect
! (OK)

consed.autoFinishHowManyTemplatesYouIntendToUseForCustomPrimerSubcloneReactions:  3
int
! this tells autofinish which templates you are planning on using
! which is necessary to figure out which regions will still be single
! subclone regions
! (OK)


consed.primersMinNumberOfTemplatesForPrimers:  1
int
! if there are fewer templates than this, the primer is rejected

// to allow for 20 bases of poor
// quality at beginning of read and then 50 bases for phrap to
// assemble together
consed.autoFinishMinBaseOverlapBetweenAReadAndHighQualitySegmentOfConsensus:  70
int
! when extending the consensus, a read that is too far from the
! consensus will not be assembled by phrap with this contig and thus
! will not be useful for extending the consensus.  This gives the
! minimum overlap of a read with the high quality segment of the
! consensus.  As reads are picked, then additional reads may be picked
! further out.
! (OK)

consed.autoFinishNumberOfVectorBasesAtBeginningOfAUniveralPrimerRead:  40
int
! used to figure out where the beginning of a reverse will be.  Not
! important to be accurate because the insert size is so uncertain
! (OK)
```

```
consed.autoFinishCDNANotGenomic:  false
bool
! If this is set to true, the whole clone is assumed to be cDNA and,
! rather than the normal method of detecting the end of the clone,
! Autofinish detects the end of the cDNA as follows:
! the user is expected to add whole read items of type 'template',
! with 'type: univ fwd' for the 5' end and 'type: univ rev' for the 3'
! end of the cDNA.
! (OK)

consed.autoFinishConfidenceThatReadWillCoverSingleSubcloneRegion:  90
int
! Autofinish computes the per cent of existing reads are aligned at
! each base position.  Typically, this number starts at around 0% at
! base position 1, rises to close to 100% at around base position 300,
! and then drops again to 0% at base position 800 or so.  This number
! specifies how high the number must be for Autofinish to consider an
! Autofinish read to cover a single subclone region.
! (OK)

consed.autoFinishPrintForwardOrReverseStrandWhenPrintingSubcloneTemplatesForCustomPrimerReads:  true
bool
! If this is true, then custom primer reads are printed out like this:
!  tccagaaaactaattcaaaataatg,56,standard.2,->,2413,2413,3681,Contig1,9,djs74_690  (fwd),10,djs74_1803  (fwd),11,djs74_1861  (fwd)
! If this is false, then custom primer reads are printed out like this:
!  tccagaaaactaattcaaaataatg,56,standard.2,->,2413,2413,3681,Contig1,9,djs74_690,10,djs74_1803,11,djs74_1861
! The difference is the (fwd) or (rev) that indicates which strand of
! the subclone template is to be used.  This is particularly important if
! you use M13 and thus must make the reverse strand.
! (OK)

consed.autoFinishPrintMinilibrariesSummaryFile:  false
bool
! If this is true, Autofinish will print a file with name
! xxx.minilibraries just as it prints one as xxx.univReverses and
!  xxx.univForwards
! (OK)

consed.autoFinishNearGapsSuggestEachMissingReadOfReadPairs:  true
bool
! This is set to true to increase the chance of closing a gap.  For
! every subclone template that has just one universal primer read
! (either just a forward or just a reverse) that might protrude off
! the end of the contig, Autofinish suggests the universal primer read
! off the opposite end of the subclone template.
! If this parameter is set false, then
! Autofinish may still choose some of these reads, but it won't
! necessarily choose them all.
! (OK)

consed.autoFinishDoNotIgnoreLCQIfThisManyBasesFromEndOfContigForLCQTagger:  300
int
! Do not ignore low consensus quality bases if they are this many
! bases from the end of the contig.
! (OK)

consed.checkIfTooManyWalks:  true
bool
! this just checks if the number of walks, pcr ends, and unknown reads
! exceeds 20% of the total number of reads.  If this is exceeded, then
! a warning message is given.  Typically, such a warning indicates
! that you have incorrectly customized determineReadTypes.perl
! (OK)

consed.numberOfColumnsBeforeReadNameInAlignedReadsWindow:  1
int
! this is for displaying information about the whole read items,
! both from PHD files and from a file

consed.compareContigsAlignsThisManyBasesMax:  2000
int
! (OK)

consed.compressedChromatExtension:  .gz
RWCString
! (OK)

consed.dimLowQualityEndsOfReads:  true
bool
!
! (OK)

consed.dimUnalignedEndsOfReads:  false
bool
! (OK)

consed.fakeReadsSpecifiedByFilenameExtension:  true
bool
! if this is true, then reads that end with .a[0-9]* or .c[0-9]* will
! be considered fake reads.  Otherwise, fake reads will be indicated
! by a WR item in the PHD file.
! (OK)

consed.fullPathnameOfAddReads2ConsedScript:  $CONSED_HOME/bin/addReads2Consed.perl
FileName
! (OK)

consed.fullPathnameOfCrossMatch:  $CONSED_HOME/bin/cross_match
FileName
! (OK)

consed.fullPathnameOfPhred:  $CONSED_HOME/bin/phred
FileName
! (OK)

consed.fullPathnameOfMiniassemblyScript:  $CONSED_HOME/bin/phredPhrap
FileName
! If you are up-to-date with phredPhrap, this script serves both
! the purpose of assemblying the entire project, as well as making
! miniassemblies.  The difference is whether phredPhrap has the
! -include_chromats option.
! (OK)

consed.gunzipFullPath:  /usr/local/bin/gunzip
RWCString
! (OK)

consed.hideSomeTagTypesAtStartup:  false
bool
! (OK)

consed.maximumNumberOfTracesShown:  4
int
! (OK)

consed.navigateAutomaticTracePopup:  false
bool
! (OK)

consed.navigateAutomaticAllTracesPopup:  false
bool
! (OK)

consed.primersMinimumLengthOfAPrimer:  15
int
! (OK)

consed.primersMaximumLengthOfAPrimer:  25
int
! (OK)

consed.primersMinimumLengthOfAPrimerForPCR:  18
int
! (OK)

consed.primersMaximumLengthOfAPrimerForPCR:  30
int
! (OK)

consed.primersMaxMeltingTempDifferenceForPCR:  3.0
double
! how large can the difference of melting temperatures be between
! two primers of a PCR primer pair
! (OK)

consed.primersMaxPCRPrimerPairsToDisplay:  100000
int
! there is a limit here, because there could possibly be millions
! (OK)

consed.primersCheckJustSomePCRPrimerPairsRatherThanAll:  true
bool
! If there are 1000 1st primers, and 1000 2nd primers, that gives
! a million pairs for Consed to check, which takes a long time.  So
! instead, just check some of the pairs
! (OK)

consed.primersNumberOfTemplatesToDisplayInFront:  2
int
! this shows the number of templates to show in the interactive primer
! picking window
! (OK)

consed.primersMaxLengthOfMononucleotideRepeat:  4
int
! (OK)

consed.primersBadLibrariesFile:  badLibraries.txt
FileName
! file of libraries, one per line
! If any template is from any one of these libraries, then
! consed/autofinish will not use this template for walking or
! suggesting any universal primer reads
! (OK)

consed.primersLibrariesInfoFile:  librariesInfo.txt
FileName
! file of libraries, with one entry for each library of the following
! format:
! LIB{
! name: library1
! avgInsertSize: 3000
! maxInsertSize: 5000
! stranded: single
! cost: 600.0
! }
! (OK)

consed.primersBadTemplatesFile:  badTemplates.txt
FileName
! file of templates that you've tried, don't work, and you don't want to try
! again
! (OK)

consed.primersChooseTemplatesByPositionInsteadOfQuality:  true
bool
! Templates for subclone custom primer walks can be chosen either on
! the basis of the quality of the template (as determined by the quality
! of existing reads from that template) or by the location of the end of
! the template.  If this parameter is false, templates will be chosen
! based solely on quality.  If this parameter is true, then templates
! with forward/reverse pairs will be picked first, followed by templates
! that have the beginning of the insert closest to the primer.
! (OK)

consed.primersWhenChoosingATemplateMinPotentialReadLength:  350
int
! when choosing templates for a custom primer, only choose a template
! if the read can be chosen at least this long
! (OK)
```

```
consed.primersWindowSizeInLookingForPCR:  2000
int
! will look this many bases back from the pointer when looking for a PCR
! primer.   Used both interactively and for Autofinish (see
!  getUnpaddedRangeForMakingPCRPrimers  )
! (OK)

consed.qualityThresholdForFindingHighQualityDiscrepancies:  40
int
! (OK)

consed.defaultVectorPathnameForRestrictionFragments:  $CONSED_HOME/lib/screenLibs/singleVectorForRestrictionDigest.fasta
FileName
! If you want to have the vector cut with the restriction
! enzymes, put the vector sequence in a file in fasta format
! and put a pathname to it here.
! (OK)

consed.fileOfAdditionalRestrictionEnzymes:
FileName
! If you want a restriction enzyme that is not in the huge list
! that comes with Consed, you can put additional enzymes in a file
! and put the full pathname of that file here.  The file must be in
! the form:
! AatI AGGCCT
! where AatI is the name of the enzyme and AGGCCT is the recognized
! sequence.  Do not include the cut site or any other information.
! There must be a single space separating them.
! (OK)

consed.commonRestrictionEnzymes:  BglII  EcoRV  NsiI  HindIII  BamHI  XhoI  PstI
RWCString
! a space-separated list of enzymes.  Make sure they match precisely
! those that are either defaults or in the file indicated by
!  consed.fileOfAdditionalRestrictionEnzymes
! (OK)

consed.defaultSelectedRestrictionEnzymes:  EcoRV  HindIII
RWCString
! a space-separated list of enzymes that will initially be
! selected when the user pops open the list of restriction enzymes.
! Currently these must be from among the consed.commonRestrictionEnzymes
! (OK)

consed.restrictionEnzymesActualFragmentsFile:  fragSizes.txt
FileName
! format like this:
! >EcoRV
! 2385
! 2489
! -1
! >XhoIII
! 259
! 3843
! -1
! (OK)

consed.restrictionDigestInitialWindowSizeInTextRows:  45
int
! (OK)

consed.restrictionDigestDoNoShowAreaOfFragmentsOverThisSize:  50000
int
! In the picture of the real and in-silico
! (OK)

consed.showReadsAlphabetically:  false
bool
! (OK)

consed.showReadsInAlignedReadsWindowOrderedByFile:  false
bool
! There are now 3 different ways to sort the reads in the Aligned
! Reads Window (top to bottom):
! 1) alphabetically in which case you should set:
!     consed.showReadsAlphabetically: true
!      consed.showReadsInAlignedReadsWindowOrderedByFile:  false
! 2) by the left end of the reads in which case you should set:
!      consed.showReadsAlphabetically:  false
!      consed.showReadsInAlignedReadsWindowOrderedByFile:  false
! 3) by a file that specifies the order of the reads in which case you
!     should set:
!      consed.showReadsAlphabetically:  false
!      consed.showReadsInAlignedReadsWindowOrderedByFile:  true
! It is an error to set:
!      consed.showReadsAlphabetically:  true
!      consed.showReadsInAlignedReadsWindowOrderedByFile:  true
! (OK)

consed.showReadsInAlignedReadsWindowOrderedByThisFile:  readOrder.txt
FileName
! This file has one read name per line.  Wildcards ('*') are allowed.
! E.g.,
! ABX*
!  myFavoriteRead.scf
! *.abi
! This means that all reads that start with ABX* will come first,
! followed by the single read myFavoriteRead.scf and then reads that end
! with .abi   A read that doesn't meet any of these criteria (e.g.,
! rs10469282 ) comes last.
! (OK)

consed.showABIBasesInTraceWindow:  false
bool
! (OK)

consed.tracesWindowInitialPixelHeight:  50
int
! (OK)
```

```
consed.assemblyViewWindowInitialPixelHeight:  500
int
! (OK)


consed.assemblyViewFileOfTemplatesToNotShow:  doNotShowInAssemblyView.fof
FileName
! (OK)

consed.assemblyViewCrossMatchMinmatch:  30
int
! value of -minmatch to be passed to crossmatch
! (OK)

consed.assemblyViewCrossMatchMinscore:  60
int
! value of -minscore to be passed to crossmatch
! (OK)

consed.assemblyViewFindSequenceMatchesForConsedScript:  $CONSED_HOME/bin/findSequenceMatchesForConsed.perl
FileName
! script that generates the file that is used by Assembly View to
! show sequence matches
! (OK)

consed.assemblyViewCrossmatchMinmatch:  50
int
! default value of -minmatch for running crossmatch with
!  findSequenceMatchesForConsed.perl
! (OK)

consed.assemblyViewCrossmatchMinscore:  50
int
! default value of -minscore for running crossmatch with
!  findSequenceMatchesForConsed.perl
! (OK)

consed.assemblyViewSequenceMatchesMinimumSimilarity:  90
int
! only show sequence matches if their simlarity is at least this
! value.  This can be changed by the user within consed/assembly view/
! by clicking on "What to show/Sequence Matches"
! (OK)

consed.tracesWindowInitialPixelWidth:  800
int
! (OK)

consed.assemblyViewWindowInitialPixelWidth:  800
int
! (OK)

consed.automaticallyScaleTraces:  true
bool
! (OK)

consed.automaticallyScaleTracesSamplePeakHeightFractionOfWindowHeight:  0.99
double
! (OK)

consed.automaticallyScaleTracesSamplePeakPercentile:  100
int
! (OK)

consed.verticalTraceMagnification:  30
int
! (OK)

consed.userDefinedKeys:  14  15
RWCString
! make a space-separated list of the decimal ASCII values of the keys
! 14 means control-N, 15 means control-O
! (OK)

consed.programsForUserDefinedKeys:  /bin/echo  /bin/echo
RWCString
! a space-separated list of the full pathnames of the commands to run
! This goes with consed.userDefinedKeys
! (OK)

consed.argumentsToPassToUserDefinedPrograms:  argument_for_first_key  argument_for_second_key
RWCString
! a space-separated list of the arguments to pass to the user-defined programs
! This goes with consed.userDefinedKeys
! (OK)

consed.tagsToApplyWithUserDefinedKeys:  none  polymorphismConfirmed
RWCString
! a space-separate list of the tag types to apply when the user
! presses a user-defined key.  If a key is to have no associated tag,
! then enter "none" for that key.
! This goes with consed.userDefinedKeys
! (OK)




consed.listOfTagTypesToHide:  matchElsewhereHighQual  matchElsewhereLowQual
RWCString
! (OK)

consed.listOfOptionalWordsToSaveInListOfReadNames:  forward  reverse  ET  BigDye  customOligo  SeqEx  FS  dyePrimer  dyeTerminator  dou
RWCString
! (OK)

consed.extendConsensusWithHighQuality:  false
bool
! When using "change consensus" to extend the consensus, make the
! read edited high quality.  This will cause phrap, the next time
! the project is assembled, to similarly extend the consensus.  If
! this is set to false, then do not change the quality of the read and
! extend the consensus with the original read qualities.
! (OK)

consed.fastStartup:  true
bool
! If you have used catPhdFiles.perl to create a huge file with all the
! xxx.phd.1 files, and you have enough memory on your computer, then
! you can startup up consed up to 7 times faster
! (OK)

consed.fastStartupFile:  phd.ball
FileName
! If you have used catPhdFiles.perl to create a huge file with all the
! xxx.phd.1 files, and you have enough memory on your computer, then
! you can startup up consed up to 7 times faster.  This file gives
! the name of the huge file.
! (OK)

consed.alwaysRunProgramToGetChromats:  false
bool
! This allows consed to get chromats out of a database, or do some
! other pre-processing of a chromat before reading it. If set to true,
! consed does not look in ../chromat_dir at all for the chromat, but
! rather runs the program listed in consed.programToRunToGetChromats
! with argument name-of-read and then reads the chromat out of
! consed.uncompressedChromatDirectory and then later deletes the
! chromat from consed.uncompressedChromatDirectory
! (OK)

consed.programToRunToGetChromats:  /usr/local/bin/myFavoriteProgram
FileName
! Set this to the program or script that you want to use to
! get a chromat and put it into /tmp (or whatever you set
! consed.uncompressedChromatDirectory to)
! (OK)

consed.autoFinishUseLongModelReadRatherThanShort:  false
bool
! When calculating the distribution of quality values at high read
! positions, should Autofinish assume that the reads that were this
! long and longer are representative of finishing reads, or should it
! assume that some finishing will not make it out this far in roughly
! the same proportion as the existing reads.
! (OK)


consed.askAgainIfWantToQuitConsedIfThisManyReads:  5000
int
! If you have to wait a long time for consed to come up, don't
! quit out of consed by mistake.
! (OK)

consed.printWindowInstructions: Make sure that the window you want to print is unobscured.   Then click "Yes" to dismiss this b
RWCString
! (OK)

consed.allowMultipleSearchForStringWindows:  false
bool
! If this is false, and there is already a SearchForString Window up,
! and the user clicks on SearchForString, it will be brought to the
! front, rather than another one being created.
! (OK)

consed.autoPCRAmplifyFalseProductsOKIfLargerThanThis:  3000
int
! If a pcr primer pair matches somewhere else and creates a product
! larger than this, the pcr primer pair will still be acceptable
! since the product will not easily form in the cycle time.
! (OK)

consed.autoPCRAmplifyMakePrimerOutOfFirstRegion:  false
bool
! I don't expect people will use this.  It allows you to amplify a
! region using autoPCRAmplify not by allowing Consed to choose each
! primer (the normal case) but rather by fixing the first primer to be
! the first area bordering the region.  I added this to allow
! non-specific priming to the transplice leader.
! (OK)

consed.autoPCRAmplifyMaybeRejectPrimerIfThisCloseToDesiredProduct:  5000
int
! --->     --->
! false    true match
! In such a case, the primer pair will be rejected if the false is
! within 5000 bases of true, even if false is a false match of the
! other primer.
!
! <---    --->
! false   true match
! In this case, the primer pair will not be eliminated.
! (OK)


consed.addNewReadsRecalculateConsensusQuality:  true
bool
! When running consed by
! consed -ace old_ace.ace -addReads fileOfPhdFiles.txt -newAceFilename new_ace.ace
! consensus quality is recalculated
! (OK)

consed.addNewReadsPutReadIntoItsOwnContig:  ifUnaligned
RWCString
! choices are:
! "always" (just put each read into its own contig)
! "ifUnaligned" (put read into a contig if it aligns against the
!     consensus, otherwise put it into its own contig)
! "never" (put read into a contig if it aligns against the consensus;
!     otherwise do not put it into the assembly)
! (OK)

consed.assemblyViewNumberOfRowsOfTags:  4
int
! (OK)



--------------------------------------------------------------------------

ACE FILE FORMAT

Refer to the accompanying sample_ace_file.txt (below)

AS <number of contigs> <total number of reads in ace file>

CO <contig name> <# of bases> <# of reads in contig> <# of base segments in contig> <U or C>

This defines the contig.  The U or C indicates whether the contig has
been complemented from the way phrap originally created it.  Thus this
is always U for an ace file created by phrap.

The contig sequence follows.  It includes pads--"*" characters which
are inserted by phrap in order to make room for some read that has an
extra base at that position.  (Note: any position which counts the *'s is
referred to as a "padded position".  A position that does not count
*'s is referred to as "unpadded position".)

BQ

This starts the list of base qualities for the unpadded consensus
bases.  (NB: annoyingly, no qualities are given for *'s in the
consensus.)  The contig is the one from the previous CO, hence no name
is needed here.


AF <read name> <C or U> <padded start consensus position>

This defines the location of the read within the contig.
C or U means complemented or uncomplemented.
<padded start consensus position> means the position of the
beginning of the read, in terms of consensus bases which start at 1
and do count *'s.

BS <padded start consensus position> <padded end consensus position> <read name>

The BS line (base segment) indicates which read phrap has chosen to be
the consensus at a particular position.

If you are writing the ace file from an assembler other than phrap,
and since most assemblers do not compute the consensus this way, you
still must write BS lines for Consed's benefit.  In this case, I
suggest you choose any read which matches the consensus perfectly over
the stretch of bases.  There must not be any two BS lines that
intersect.  Each unpadded base must be included in some BS line.

RD <read name> <# of padded bases> <# of whole read info items> <# of read tags>
Below RD is the sequence of bases for the read.  The sequence includes
*'s and is in the orientation that phrap needed to align it against
the consensus (thus it might be complemented from the direction it was
sequenced).

QA <qual clipping start> <qual clipping end> <align clipping start> <align clipping end>

This line indicates which part of the read is the high quality segment
(if there is any) and which part of the read is aligned against the
consensus.  These positions are offsets (and count *'s) from the left
end of the read (left, as shown in Consed).  Hence for bottom strand
reads, the offsets are from the end of the read.  The offsets are
1-based.  That is, if the left-most base is in the aligned,
high-quality region, <qual clipping start> = 1 and <align clipping
start> = 1 (not zero).  If the entire read is low quality, then <qual
clipping start> and <qual clipping end> will both be -1.

DS CHROMAT_FILE: <name of chromat file> PHD_FILE: <name of phd file> TIME: <date/time of the phd file> CHEM: <prim, term, unk

This line must contain information that matches the phd file.  If you
are writing an ace file, pay particular attention to this line.  Make
sure that Consed can read your ace file without reporting any errors.

There can be additional information on this line.
This replaces the DESCRIPTION line from the old ace file.

The following is for transient read tags (those generated by
crossmatch and phrap).

RT{
<read name> <tag type> <what program created tag> <padded read pos start> <padded read pos end> <date when tag was created ir
}

for example:

RT{
djs14_680.s1 matchElsewhereLowQual phrap 904 933 990823:114356
}

There are consensus tags now in the ace file.  All consensus tags have
the following format:

CT{
<contig name> <tag type> <what program created tag> <padded cons pos start> <padded cons pos end> <date when tag was created
(possibly additional information)
}

The NoTrans is optional--it indicates that, when you reassemble, this
tag should not be transferred to the new assembly.  This is true with
tags that should be recreated each time because they have to do with
the assembly (e.g., repeat tags).
```

```
e.g.,

CT{
Contig206 repeat tagRepeats.perl 118732 119060 990823:115033 NoTrans
AluY
}
```

In the case of most consensus tag types, there is only 1 line for the consensus tag. In the case of comment tags and oligo tags, there are additional lines of information. The comment tag includes the comment on the additional lines. The oligo tag has the following information: <oligo name> <oligo bases from 5' to 3'> <melting temp> <C or U indicating whether the oligo is top strand or bottom strand relative to the orientation of the contig as created by phrap>

```
WA{
<tag type> <what program created tag> <date tag was created in form YYMMDD:HHMISS>
1 or more lines of data
}
```

This line is a 'whole assembly' tag. It is used for information referring to the assembly as a whole. Currently, phrap puts its version and phrap command line options in a WA tag.

You can append CT, WA, and RT tags to the end of the ace file in any order you like.

Sample Ace File:

```
AS 1 8

CO Contig1 1475 8 156 U
agccccgggccgtggggttccttgagcactcccaaagttccaacccagga
tgtccccgacgcttaaaccttccaagtctgaaacgggaaatttgatttgc
gggctaggataaacgccggggagaaaggcagaactgccttttacccccca
aggatatcccttgggaagggcccctttgcactcagctgctccctaattat
ggcgatcctccctctatctttgtcccccctgtctttcaggatccctctcAA
CAACAgaccaCTCccattaaaGAAATCtccttctgatctgcgggatcACA
TAAAACAGTGCCattcAAaAcgtcccttcCcccAATGTCtaagtgTggtg
gagcCcttcctgcCCggctctgtgcacccacggtgcctgcatgaccccgg
atGCAGTGTGCACCAGctCCCATCATTCAAgagCATGACTGTGTTGCCAA
CCAGCcacCAGGCACTGGGGAGGGAGCtgaGGGAGCAcaaAAGGGATGAG
CCACCCTCTGTcCcagAAGTGGAGGGCATGGGGCTTGGCTGGGCTTAGAG
CTAACATACACAGGATGCTGAAAAAGAACAACACAAggtGTGTGGAGCAA
AGGAAAGGGAAATCAGCTTGAAGCTGATGTTAGTGTGCTTGGGCTGAGTA
CAGCCATGCTCTCAGTTGAGGCACGGTTGGCTCCCCATGGGCAAGATCCC
TCCTGGCCCATCTCTCCTCTTATTCTCTATCCCTTCCCCAGGTCCCTGCC
TTAGAGGTTTCACCAGAGCACAGCTCCTGCCTGTGGCCAAAACAGTATTT
GGCCACTCACCGACCCAGTGTCAGC*ATCCAGATGGGTTCCACATCTCAC
AACCCT*GAGCAGCAGAGAAGGGTTTGAAAGGCCAGGGGAG*AATGAAGA
CGAAGGAGG*TGTTGGCAACAACACAGA*G*AGTCAGCAGCCAGAACGCC
AGGTATCCACACACATAAGACATTCTAAATTTTTACTCAACAGAAATTGT
CTATGTCTGTGTCTGGGCACCATGGCAACACCTTATCTCTACAAAAATTA
GCGGAATGTAGTGGTGCCTGTGTGTAGTCCCAGCTATTCAAGAGGCTGAA
GTGGGAGGATTGCTTGAGCCATGGAAGTCAAGGCTGTAGTGAGCCATGAT
TGTGTCAATGCACTCCAGACAGAGCAAGACCCTGCTCCCACCACACACCT
CaaacgaaAAAAAAaaagggcaaagatatgaactgaaatggaatatag*a
gcagcaaaaggaacagaaaattgtctatgcctggttctctagtcatgtgc
agaacagacagtatcccggccctattgagttcttggggcagttaggcttg
tgcacccttgcttctatgccacagttagggcattcgggattcccatcctt
ttccccggggttgctttttgtttgcgattaccttttcggaacaatggggg
gaaattattttccaagttgggtttg


BQ
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 22
 23 26 24 25 25 17 17 13 14 19 21 22 22 17 17 11 8 7 10 13 18 23 28 28 31 31 32 18 18 10 10 10 12 15 8 6 6 8 8 10 15 9 11 12
 30 31 24 24 22 24 25 28 23 27 24 27 18 15 15 16 21 23 18 20 13 8 7 7 12 10 9 10 10 21 12 14 14 28 27 32 24 23 20 19 15 17 15
 14 10 10 10 23 10 10 10 10 10 11 11 18 25 24 10 10 10 10 10 14 10 11 11 11 13 12 12 10 12 10 10 10 10 10 10 14 10 12 10 10 1
 17 19 24 32 37 37 37 37 32 30 30 30 28 23 23 25 15 15 20 27 32 23 22 22 27 32 34 34 21 21 12 12 12 24 32 41 45 45 37 45 45 4
 37 37 41 32 32 14 14 19 32 28 37 37 41 41 45 45 37 37 37 30 30 32 32 37 37 32 28 16 16 17 32 32 37 45 37 25 25 9 9 9 25 25 3
 37 37 45 45 37 37 37 37 38 25 25 12 25 10 10 15 32 47 52 62 62 55 43 43 34 43 43 58 58 78 77 72 72 70 70 70 74 77 69 68 55 5
 64 58 56 56 64 65 67 70 70 75 79 70 70 70 70 70 70 67 71 71 71 84 63 63 62 62 62 59 59 61 61 64 64 49 42 32 10 6 18 32 35 46
 55 49 46 47 47 55 55 55 54 47 47 47 48 48 54 54 54 48 48 55 47 47 47 55 49 48 48 48 55 47 48 48 47 47 47 46 48 48 48 50 44 4
 74 66 66 58 54 60 68 68 61 63 47 57 45 74 85 78 70 65 62 61 61 55 73 65 59 61 75 77 80 86 81 81 83 85 85 85 90 84 78 78 73 7
 87 78 72 75 72 72 76 79 82 88 90 89 89 89 89 89 90 90 90 85 85 79 83 83 90 90 90 90 90 90 90 90 90 90 90 90 89 89 89 90 9
 90 90 90 81 66 66 62 62 62 73 89 90 90 86 86 86 86 88 88 90 90 90 90 90 90 90 88 71 68 61 61 66 66 70 65 64 70 70 76 90 90 9
 87 79 79 79 79 89 74 65 71 72 79 73 73 70 75 79 76 81 81 83 80 87 89 90 82 82 90 88 88 88 88 89 86 77 77 80 79 79 79 90 90 9
 57 65 76 76 76 80 89 89 89 90 90 90 90 88 88 88 88 88 88 90 90 90 90 90 90 90 90 90 90 88 90 90 90 90 90 90 90 90 90 90 90 9
 45 68 70 61 75 76 73 68 84 88 90 90 90 90 90 89 72 54 62 62 53 55 55 80 83 80 80 83 85 83 87 83 83 83 85 85 86 86 84 81 83 8
 88 87 90 90 90 90 85 84 82 71 75 62 62 37 68 75 77 74 70 71 70 72 72 80 80 80 84 83 82 66 70 55 55 55 37 55 55 55 55 55 55 5
 47 47 47 47 47 47 47 47 55 50 50 50 47 47 47 47 44 44 55 48 51 51 54 54 54 54 54 55 54 54 55 55 55 55 55 55 55 55 55 55 55 5
 61 61 61 61 44 42 34 34 37 37 37 44 47 47 47 61 61 61 61 61 61 61 47 49 48 47 55 54 55 55 55 55 55 44 44 44 44 46 43 43 44 4
 44 44 39 39 43 42 50 42 42 38 37 38 41 50 52 55 47 47 39 44 44 46 41 42 40 43 40 41 42 38 37 42 55 50 44 44 46 48 55 55 55 3
 42 55 46 46 46 48 47 48 46 43 41 39 42 39 44 44 44 48 48 38 36 36 38 38 38 44 44 44 44 44 44 42 42 36 41 40 36 36 30 33 32 2
 13 14 23 20 21 28 28 31 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

AF K26-217c U 498
AF K26-526t U 510
AF K26-961c U 577
AF K26-394c U 797
AF K26-291s U 828
AF K26-822c U 883
AF K26-572c C 1
AF K26-766c C 408
BS 1 515 K26-572c
BS 516 516 K26-217c
BS 517 521 K26-572c
BS 522 529 K26-217c
BS 530 538 K26-572c
BS  539  569  K26-217c
BS  570  571  K26-526t
BS  572  573  K26-217c
BS  574  579  K26-526t
BS  580  584  K26-217c
BS  585  591  K26-526t
BS  592  592  K26-217c
BS  593  601  K26-526t
BS  602  604  K26-217c
BS  605  606  K26-526t
BS  607  607  K26-217c
BS  608  621  K26-526t
BS  622  628  K26-217c
BS  629  629  K26-526t
BS  630  630  K26-217c
BS  631  633  K26-526t
BS  634  634  K26-217c
BS  635  635  K26-526t
BS  636  639  K26-217c
BS  640  646  K26-526t
BS  647  648  K26-217c
BS  649  649  K26-526t
BS  650  650  K26-217c
BS  651  654  K26-766c
BS  655  655  K26-961c
BS  656  656  K26-217c
BS  657  669  K26-961c
BS  670  675  K26-217c
BS  676  676  K26-961c
BS  677  688  K26-217c
BS  689  693  K26-526t
BS  694  696  K26-217c
BS  697  698  K26-526t
BS  699  700  K26-961c
BS  701  706  K26-217c
BS  707  707  K26-961c
BS  708  708  K26-217c
BS  709  709  K26-961c
BS  710  710  K26-526t
BS  711  775  K26-961c
BS  776  776  K26-766c
BS  777  777  K26-961c
BS  778  834  K26-766c
BS  835  837  K26-961c
BS  838  840  K26-394c
BS  841  882  K26-766c
BS  883  884  K26-394c
BS  885  898  K26-766c
BS  899  899  K26-961c
BS  900  900  K26-766c
BS  901  901  K26-961c
BS  902  934  K26-766c
BS  935  935  K26-394c
BS  936  936  K26-766c
BS  937  937  K26-394c
BS  938  940  K26-766c
BS  941  944  K26-394c
BS  945  945  K26-291s
BS  946  948  K26-822c
BS  949  949  K26-766c
BS  950  951  K26-822c
BS  952  954  K26-766c
BS  955  955  K26-822c
BS  956  957  K26-394c
BS  958  962  K26-822c
BS  963  963  K26-394c
BS  964  970  K26-822c
BS  971  971  K26-394c
BS  972  972  K26-822c
BS  973  973  K26-394c
BS  974  976  K26-822c
BS  977  979  K26-394c
BS  980  986  K26-291s
BS  987  987  K26-394c
BS  988  1004  K26-822c
BS  1005  1009  K26-394c
BS  1010  1012  K26-291s
BS  1013  1014  K26-394c
BS  1015  1021  K26-822c
BS  1022  1022  K26-394c
BS  1023  1026  K26-822c
BS  1027  1028  K26-291s
BS  1029  1036  K26-822c
BS  1037  1052  K26-291s
BS  1053  1053  K26-822c
BS  1054  1060  K26-291s
BS  1061  1061  K26-822c
BS  1062  1062  K26-291s
BS  1063  1065  K26-394c
BS  1066  1068  K26-822c
BS  1069  1079  K26-291s
BS  1080  1081  K26-822c
BS  1082  1082  K26-291s
BS  1083  1084  K26-822c
BS  1085  1089  K26-291s
BS  1090  1094  K26-822c
BS  1095  1096  K26-394c
BS  1097  1099  K26-822c
BS  1100  1100  K26-291s
BS  1101  1104  K26-822c
BS  1105  1105  K26-394c
BS  1106  1110  K26-822c
BS  1111  1115  K26-291s
BS  1116  1122  K26-822c
BS  1123  1124  K26-291s
BS  1125  1135  K26-822c
BS  1136  1136  K26-394c
BS  1137  1139  K26-822c
BS  1140  1140  K26-291s
BS  1141  1150  K26-822c
BS  1151  1155  K26-291s
BS  1156  1161  K26-822c
BS  1162  1164  K26-291s
BS  1165  1167  K26-822c
BS  1168  1173  K26-291s
BS  1174  1175  K26-822c
BS  1176  1189  K26-291s
BS  1190  1196  K26-822c
BS  1197  1199  K26-291s
BS  1200  1221  K26-822c
BS  1222  1225  K26-291s
BS  1226  1227  K26-822c
BS  1228  1228  K26-394c
BS  1229  1231  K26-291s
BS  1232  1233  K26-822c
BS  1234  1235  K26-291s
BS  1236  1236  K26-394c
BS  1237  1239  K26-291s
BS  1240  1242  K26-822c
BS  1243  1244  K26-291s
BS  1245  1247  K26-394c
BS  1248  1255  K26-822c
BS  1256  1256  K26-291s
BS  1257  1257  K26-394c
BS  1258  1258  K26-291s
BS  1259  1259  K26-822c
BS  1260  1260  K26-394c
BS  1261  1265  K26-291s
BS  1266  1266  K26-822c
BS  1267  1268  K26-394c
BS  1269  1269  K26-822c
BS  1270  1275  K26-291s
BS  1276  1280  K26-822c
BS  1281  1281  K26-394c
BS  1282  1290  K26-822c
BS  1291  1292  K26-291s
BS  1293  1294  K26-822c
BS  1295  1297  K26-291s
BS  1298  1301  K26-822c
BS  1302  1302  K26-291s
BS  1303  1475  K26-822c

RD  K26-217c  563  0  0
tcccCgtgagatcatcctgaAGTGGAGGGCATGGGGCTTGGCTGGGCTTA
GAGCTAACATACACAGGATGCTGAAAAAGAACAACACAAgntGTGTGGAG
CAAAGGAAAGGGAAATCAGCTTGAAGCTGATGTTAGTGTGCTTGGGCTGA
GTACAGCCATGctntCAGTTGAGGCACGGTTGGCTCCCCATGGGCAAGAT
CCCTCCTGGCCCATCTCTCCTCTTATTCTCTATCCCTTCCCCAGGTCCCT
GCCTTAGAGGTTTCACCAGAGCACAGCTCCTGcctgtggccaAAACAGTA
TTTGGCCACTCACcGAcccagTGTCAGC*atccaGatggGtTccacatct
cacaaccct*gggcagcagagaaggggtttaaaggccagggggg*tatta
agccgaaggagg*ttttggaaacaccaaggg*g*ggtcagaccccaacgc
cagtttccccaaaaaggggcattcaaatttttttctcagagattttcttt
cctttttgggccccgggaaccttttttaaaaaatgggggattgggcccc
cttggccccctc

QA  19  349  19  424
DS  CHROMAT_FILE:  K26-217c  PHD_FILE:  K26-217c.phd.1  TIME:  Thu  Sep  12  15:42:38  1996
RD  K26-526t  687  0  0
ccgtcctgagtggAGggcatggggcttggctggGCTTAGAGCTAACATAC
ACAGGATGCTGAAAAAGAACAACACAAggtGTGTGGAGCAAAGGAAAGGG
AAATCAGCTTGAAGCTGATGTTAGTGTGCTTGGGCTGAGTACagcnatgc
tntgaGTTGAggaacgGTTGGCTCCCCATGGGCAAGATCCCTCCTGGCCC
ATCTCTCCTCTTATTCTCTATCCCTTCCCCAGGTCCCTGCCTTAGAGGTT
TCACcAgAGCACAgCTCctgcctgtggccaAAACAGTATTTGGccACTCA
CCGAcCCAGTGTcagt*atccAGATGGGttccACATCtcacagcccT*Ga
gcAgcagngaaGGGTttgaaagggcAggggggggaatgaaGacggaggagg
gtgttggcaaccacacaga*ggagtcaggaggcaggacggcaggtatccA
Cacacattaggcattttaaatttttacttaacaggaattgtctatggctg
ggtttgggaac*atgggaacacctattcttt*caaaa*ttggggggat*t
agtggtgc*tgt*tatagtcccgttattaaGggttaagtggggtttcttt
gccaggaggtaaggtttggggccctatttttaattacttggaaggaagcc
ttttcccagataaggaaaaaggaggtTTtttgtttta

QA  12  353  9  572
DS  CHROMAT_FILE:  K26-526t  PHD_FILE:  K26-526t.phd.1  TIME:  Thu  Sep  12  15:42:33  1996
RD  K26-961c  517  0  0
aatattaccggcgcggggttCcgTCGGAAAGGGAAATCAGCTTGAAGCTG
ATGTTAGTGTGCTTGgGCTGAGTacaGCCATGCTCTCAGTTGAGGCACGG
TTGGCTCCCCATGGGCAAGATCCCTCCTGGCCCATCTCTCCTCTTATTCT
CTATCCCTTCCCCAGGTCCCTGCCTTAGAGGTTTCACCAGAGCACAGCTC
CTGccTGTGGCCAAAACAGTATTTGGccactgaccGACCCagtGTCAGC*
ATCCAGATGGGTTCCACATCTCacaaccCT*GAGCAGCAGAGAAGGGTTT
GAaagGcCAGGGGAG*AATGAAGACgaaggaGG*TGTTgGcaacaacaca
gA*G*AGTCAGCAGccAgaacgccaggtatccacACACATaaggCATtct
aaatttttaCtcaACaggaattgtctATgtctgtgTCtgggcaccagggc
a*cacctTATCTCTAcaaaaat*agcgggatttagtggtgcttgtgtg**
g*cccagctattcaggg

QA  20  415  26  514
DS  CHROMAT_FILE:  K26-961c  PHD_FILE:  K26-961c.phd.1  TIME:  Thu  Sep  12  15:42:37  1996
RD  K26-394c  628  0  0
ctgcgtatcgtcacc*accCAGTGTCagctatcCAGATGGGTTCCACATC
TcacaacCCT*GAGCAGCAGAGAAGGGTTTGAAAGGCCAGGGGAG*AATG
AAGACga*gGAGG*tgTTGGCAACAacacagA*G*AGTCAGCAGCCAGAA
CGCCAGGTATCCACACACATAAGACATTCTAAATTTTTACTCAACAGAAA
TTGTCTATGTCTGTGTCTGGgcaCCATGGCAACACCTTATCTCTACAAAA
ATTAGCGGAATGTAGTGGTGCCTGtgtGTAGTCCCAGCTATTCaaGAGGC
TGAAGTGGGAGGATTGCTTGagccaTggaagtcaagGCTGTAGTGagCCa
TGattgtgtCaATGCACtcnagAcagagcaaGACCctgctcccaccacac
aacttaanaggaaaaaaaaaaaggaaaagaaatgaaatgaaatgggatat
ag*aa*aggaaaagga*cagaaa*ttgtctatgcctggt*ctctagtaat
gtcagtcagccagtttccagccttttggtcttgggcattctgctgtcaca
atctcttggaacgttgggcagggaatcccatttttcccccgtttTttttt
gtggcaattaccttttggaaccctgggt

QA  18  368  11  502
DS  CHROMAT_FILE:  K26-394c  PHD_FILE:  K26-394c.phd.1  TIME:  Thu  Sep  12  15:42:32  1996
RD  K26-291s  556  0  0
gaggatcgcttTCCacatctcaCAaccctcgagCAgCagagAAgggTTTG
```

```
AAAGGCCAGGGGAG*AATGAAGACGa*ggAGG*TGTTGGCAACAacacag
a*G*AGTCAGCAGCCAGAACGCCAggtaTCCAcacacataAgccatTCTA
AATTTTTACTCAAcagAAATTGTCTAtgTCTGTGTCTGggcacCATGGCA
ACACCTTATCTCTACAAAAATTAGCGGAATGTAGTggtGCCTGTGTGTAG
TCCCAGCTATTCAAgaggctGAAGTgcgaggatTGCTTgagCCATGGAAG
TcaaggctgtAGTGAgccatgatTGTGTCAATGCACTCCAGACAGAGCAA
GACCCTGCTCCCAccaCACAcctcaaaaggtattgattaaaGGAaAagaa
atgaaAtgaaatgagataaaggaaaaggaaaaagaacaggatattgTCtA
Tgcctgat*ctctagt*atgtgcagacagaagtttccagccactgagttc
ttgccccagctaactttttacaaatccccctggggaaggtttggcccagg
cagatg

QA 11 373 11 476
DS CHROMAT_FILE: K26-291s PHD_FILE: K26-291s.phd.1 TIME: Thu Sep 12 15:42:31 1996
RD K26-822c 593 0 0
ggggatccg*tcatgagacga*ggAGG*TGTTGGCAACa*ca*agaag*A
GTCAGCAGCCAGAACGCCAGGTATCCACACACATAAGACATTCTAAATTT
TTACTCAACAGAAATTGTCTATGTCTGtgtCTGGGCACCATGGCAACACC
TTATCTCTACAAAAATTAGCGGAATGTAGTggTGCCTGtgtGTAGTCCCA
GCTATTCAAGAGGCTGAAGTGGGAGGATTGCTTGAGCCATGGAAGTCAAG
GCTGTAGTGAGCCATGATTGtgtCAATGCACTCCAGAcAgAGCaAgacCC
tgCTCccACCACACacctCaaacgaaAAAAAAaaagggcaaagatatgaa
ctgaaatggaatatag*agcagcaaaaggaacagaaaattgtcTATGcct
ggttctctagtcatgtgcagaacagacagtatcccggccctattgagttc
ttggggcagttaggcttgtgcacccttgcttctatgccacagttagggca
ttcgggattcccatcctttccccggggttgctttttgtttgcgattacc
ttttcggaacaatgggggaaattattttccaagttgggtttg

QA 25 333 16 593
DS CHROMAT_FILE: K26-822c PHD_FILE: K26-822c.phd.1 TIME: Thu Sep 12 15:42:36 1996
RD K26-572c 594 0 0
agccccgggccgtggggttccttgagcactcccaaagttccaacccagga
tgtccccgacgcttaaaCcttccaagtctgaaacgggaaAtttgatttgc
gggctaggataaacgccggggagaaaggcagaactgccttttaccCCcca
aggatatcccttgggaagggcccctttgcactcagctgctccctaattat
ggcgatcctccctctatctttgtccccctgtctttcaggatccctctcAA
CAACAgaccaCTCccattaaaGAAATCtccttctgatctgcgggatcACA
TAAAACAGTGCCattcAAaAcgtcccttcCcccAATGTCtaagtgTggtg
gagcCcttcctgcCCggctctgtgcacccacggtgcctgcatgaccccgg
atGCAGTGTGCACCAGctCCCATCATTCAAgagCATGACTGTGTTGCCAA
CCAGCcacCAGGCACTGGGGAGGGAGCtgaGGGAGCAcaaAAGGGATGAG
CCACCCTCTGTcCcagAAGTGGAgcgcATGGGGCTTGGCTgggcTTAGAG
CtaacaTACACAGGATGCTGAAaaagaaCAACACaatagtaaca

QA 249 584 1 586
DS CHROMAT_FILE: K26-572c PHD_FILE: K26-572c.phd.1 TIME: Thu Sep 12 15:42:34 1996
RD K26-766c 603 0 0
gaataattggaatcacggcaaaaatttggggacaaatattatttccaaaa
ttcccccagcaatcacacaggccctcaagcccatcaactcggtcattcac
cgattttcctaaatcaagggtattagcttg*ctgggcttacacctaacat
acacagcatgctcaatgagaAcaatacgagctgtgtggagcacaggaagg
ggaAAtcagcctgaagctgctgttagtgtgcttgg*ctgAGTACAGCcaT
GCTctCAGTTgaggcAcggTTGGCTCCCCATGGgCAAGATCCCTCCTggC
CCATCTCTCCTCTTaTTCTCTATCCCTTCCCCAGGTCCCTGCCTTAGagg
tttCACCAGAGCACAGCTCCTGCCTGTGGCCAAAACAGTATTTGGCCACT
CACCGACCCAGTGTCAGC*ATCCAGATGGGTTCCACATCTCACAACCCT*
GAGCAGCAGAGAAGGGTTTGAAAGGCCAGGGGAG*AATGAAGACGAAGGA
GG*TGTTGGCAACAACACAGA*G*AGTCAGCAGCCAGAACGCCAGGTATC
CACACACATAagaCATtctaAATTTTTACTCAAacgatcCccggaaccac
acg

QA 240 584 126 583
DS CHROMAT_FILE: K26-766c PHD_FILE: K26-766c.phd.1 TIME: Thu Sep 12 15:42:35 1996

WA{
phrap_params phrap 990621:161947
/usr/local/genome/bin/phrap standard.fasta.screen -new_ace -view
phrap version 0.990319
}

CT{
Contig1 repeat consed 976 986 971218:180623
}

CT{
Contig1 comment consed 996 1007 971218:180623
This is line 1 of a comment
There may be any number of lines
}

CT{
Contig1 oligo consed 963 987 971218:180623
standard.1 acataagacattctaaattttact 50 U
seq from clone
}
```

Prior to 1998 there was a different ace file format. If you still haven't changed to the new ace file format, you must do so now since it contains information that is not contained in the old ace file format. This additional information (e.g., the alignment and quality clipping values) are essential for some of the Consed functions (e.g., navigate by single stranded, navigate by single subclone, Autofinish) to work correctly.

Another reason to switch to the new ace format is that you will get faster Consed startup performance. The new ace file format is also much smaller (about 60% as big as the old).

The new phrap (Aug 1998 and better) writes the new ace format (using the -new_ace or -ace switch). Since Consed now uses the additional information found only in the new ace format, if you are editing an assembly, you should first re-phrap to take advantage of this additional information.

Consed use with old ace format is no longer supported.

WHAT THE COLORS MEAN

See the beginning of the Quick Tour (above). But here is a very partial list of the colors:

```
Greyscale of background indicates quality
Grey base with black background--clipped off part of read (either due
     to low quality or due to alignment)
Red base--discrepant with consensus
Black base--agrees with consensus
Colored area covering half of a base--tag (see Quick Tour)
Purple tag--more than 1 tag covering a base
```