lepbase-analysis Documentation

Release 1

Richard Challis, Sujai Kumar

May 12, 2016

Contents:

CHAPTER 1

Repeat masking

Contents:

1.1 BuildDatabase

Wrapper around makeblastdb to format blast databases for use with *RepeatModeler*.

1.1.1 Program parameters

```
-name <database name>
    The name of the database to create.

-engine <engine name>
    The name of the search engine we are using. I.e abblast/wublast or
    ncbi (rmblast version).

-dir <directory>
    The name of a directory containing fasta files to be processed. The
    files are recognized by their suffix. Only *.fa and *.fasta files
    are processed.

-batch <file>
    The name of a file which contains the names of fasta files to
    process. The files names are listed one per line and should be fully
    qualified.
```

1.1.2 Typical usage

```
/exports/software/repeatmodeler/RepeatModeler/BuildDatabase -engine ncbi -name mydb seq.fa
```

1.1.3 Gridengine wrapper

See *RepeatModeler*.

1.2 RepeatModeler

RepeatModeler (link)

version 1.08

de novo repeat family identification and modeling using RECON, RepeatScout, *TRF* & RMBlast *[0] on blast databases formatted with *BuildDatabase*.

1.2.1 Program parameters

```
-database
    The prefix name of a XDF formatted sequence database containing the
    genomic sequence to use when building repeat models. The database
    may be created with the WUBlast "xdformat" utility or with the
    RepeatModeler wrapper script "BuildXDFDatabase".

-engine <abblast|wublast|ncbi>
    The name of the search engine we are using. I.e abblast/wublast or
    ncbi (rmblast version).

-pa #
    Specify the number of shared-memory processors available to this
    program. RepeatModeler will use the processors to run BLAST searches
    in parallel. i.e on a machine with 10 cores one might use 1 core for
    the script and 9 cores for the BLAST searches by running with "-pa
    9".

-recoverDir <Previous Output Directory>
    If a run fails in the middle of processing, it may be possible
    recover some results and continue where the previous run left off.
    Simply supply the output directory where the results of the failed
    run were saved and the program will attempt to recover and continue
    the run.
```

1.2.2 Typical usage

```
/exports/software/repeatmodeler/RepeatModeler/RepeatModeler -database mydb -engine ncbi -pa 32
```

1.2.3 Gridengine wrapper

RepeatModeler will use more threads than defined by `-pa` so use `nice`

repeatmodeler.sh:

```
#!/bin/bash

#$ -V
#$ -cwd
#$ -j y
#$ -o $JOB_ID.log

# define $DATADIR and $DATABASE before/when calling this script, e.g.:
```

[0] Alternatively ABBlast/WUBlast may be used.

```
#       export DATADIR=`pwd`
#       qsub -v SEQFILE=seq.fa -v DATABASE=mydb ...

WORKDIR=/scratch/$USER/repeatmodeler/$JOB_ID
mkdir -p $WORKDIR
cp $DATADIR/$SEQFILE $WORKDIR
cd $WORKDIR
/exports/software/repeatmodeler/RepeatModeler/BuildDatabase -engine ncbi -name $DATABASE $SEQFILE
nice -n 10 /exports/software/repeatmodeler/RepeatModeler/RepeatModeler -database $DATABASE -engine nc
rsync -a ./consensi.fa.classified $DATADIR/$SEQFILE.consensi.fa.classified
```

1.2.4 qsub command

```
export DATADIR=`pwd`
qsub -v SEQFILE=seq.fa -v DATABASE=mydb -pe smp 32 /path/to/repeatmodeler.sh
```

1.2.5 parallel qsub command

Submit a job for each `.fa` file in a directory, limit to 2 simultaneous jobs to avoid flooding queue.

```
export DATADIR=`pwd`
parallel -j 2 qsub -v SEQFILE={} -v DATABASE={.} -pe smp 32 -sync y /path/to/repeatmodeler.sh ::: *.f
```

1.2.6 Footnotes

1.3 Filter RepeatModeler Library

Repeat libraries built with *RepeatModeler* may contain non-TE protein coding sequences. If masking an annotated genome, filter the library using *RepeatMasker* `RepeatPeps.lib` and a proteome to remove 'genuine' protein hits.

1.3.1 Filtering steps

Blast proteome against RepeatMasker TE database

```
blastp -query proteins.fa \
       -db /exports/software/repeatmasker/repeatmasker-4-0-5/Libraries/RepeatPeps.lib \
       -outfmt '6 qseqid staxids bitscore std sscinames sskingdoms stitle' \
       -max_target_seqs 25 \
       -culling_limit 2 \
       -num_threads 48 \
       -evalue 1e-5 \
       -out proteins.fa.vs.RepeatPeps.25cul2.1e5.blastp.out
```

Remove TEs from proteome

```
fastaqual_select -f transcripts.fa \
       -e <(awk '{print $1}' proteins.fa.vs.RepeatPeps.25cul2.1e5.blastp.out | sort | uniq) > transcr
```

Blast proteome against RepeatModeler library

```
makeblastdb -in transcripts.no_tes.fa -dbtype nucl
blastn -task megablast \
       -query consensi.fa.classified \
       -db transcripts.no_tes.fa \
       -outfmt '6 qseqid staxids bitscore std sscinames sskingdoms stitle' \
       -max_target_seqs 25 \
       -culling_limit 2 \
       -num_threads 48 \
       -evalue 1e-10 \
       -out repeatmodeller_lib.vs.transcripts.no_tes.25cul2.1e10.megablast.out
```

Remove hits from RepeatModeler library

```
fastaqual_select -f consensi.fa.classified \
       -e <(awk '{print $1}' ../../repeatmodeller_lib.vs.transcripts.no_tes.25cul2.1e25.megablast.out
```

1.3.2 Grid engine wrapper

filter_repmodlib.sh:

```
#!/bin/bash

#$ -V
#$ -cwd
#$ -j y
#$ -o $JOB_ID.log
. /etc/profile.d/modules.sh
module load blast

# export DATADIR=/path/to/
# qsub -v PROTSEQFILE=proteins.fa
#       -v BLASTPDB=/path/to/RepeatPeps.lib
#       -v TSCSEQFILE=transcripts.fa
#       -v REPMODLIB=consensi.fa.classified

WORKDIR=/scratch/$USER/blastp/$JOB_ID
mkdir -p $WORKDIR
cd $WORKDIR

# Blast proteome against RepeatMasker TE database
blastp -query $DATADIR/$PROTSEQFILE -db $BLASTPDB -outfmt '6 qseqid staxids bitscore std sscinames s
       -max_target_seqs 25 -culling_limit 2 -num_threads $NSLOTS -evalue 1e-5 \
       -out $PROTSEQFILE.vs.RepeatPeps.25cul2.1e5.blastp.out

# Remove TEs from proteome
/exports/colossus/lepbase/repeatmasking/fastaqual_select -f $DATADIR/$TSCSEQFILE \
       -e <(awk '{print $1}' $PROTSEQFILE.vs.RepeatPeps.25cul2.1e5.blastp.out | sort | uniq) > $TSCSE

# Blast proteome against RepeatModeler library
makeblastdb -in $TSCSEQFILE.no_tes.fa -dbtype nucl
blastn -task megablast \
       -query $DATADIR/$REPMODLIB \
       -db $TSCSEQFILE.no_tes.fa \
       -outfmt '6 qseqid staxids bitscore std sscinames sskingdoms stitle' \
       -max_target_seqs 25 \
       -culling_limit 2 \
       -num_threads $NSLOTS \
       -evalue 1e-10 \
       -out $REPMODLIB.vs.$TSCSEQFILE.25cul2.1e10.megablast.out

# Remove hits from RepeatModeler library
/exports/colossus/lepbase/repeatmasking/fastaqual_select -f $DATADIR/$REPMODLIB \
       -e <(awk '{print $1}' $REPMODLIB.vs.$TSCSEQFILE.25cul2.1e10.megablast.out | sort | uniq) > $RE

rsync -a --remove-source-files ./$REPMODLIB.filtered_for_CDS_repeats.fa $DATADIR
```

qsub command

```
export DATADIR=`pwd`
qsub -pe smp 16 -v PROTSEQFILE=proteins.fa -v BLASTPDB=/path/to/RepeatPeps.lib -v TSCSEQFILE=transcri
```

1.4 Combine Repeat Libraries

Use the *RepeatMasker* tool `queryRepeatDatabase.pl` to extract taxon-specific repeats:

```
/exports/software/repeatmasker/repeatmasker-4-0-5/util/queryRepeatDatabase.pl \
        -species "taxon" > repeatmasker.taxon.fa
```

and combine with a repeatmodeller library. The output of `queryRepeatDatabase.pl` has an additional line at the begining before the start of the fasta format sequences so use `tail -n+2` to skip the first line:

```
tail -n+2 repeatmasker.taxon.fa | cat - taxon.consensi.fa.classified > taxon.repeatlib.fa
```

1.5 RepeatMasker

1.5.1 Typical Usage

```
/exports/software/repeatmasker/repeatmasker-4-0-5/RepeatMasker \
        -pa 48 \
        -lib taxon.repeatlib.fa \
        -dir . \
        -xsmall \
        seq.fa
```

1.5.2 Gridengine wrapper

repeatmasker.sh:

```
#!/bin/bash

#$ -V
#$ -cwd
#$ -j y
#$ -o $JOB_ID.log

#       export DATADIR=`pwd`
#               -v SEQFILE=seq.fa
#               -v REPLIB=repeatLib.fa ...

WORKDIR=/scratch/$USER/repeatmasker/$JOB_ID
mkdir -p $WORKDIR
cd $WORKDIR
/exports/software/repeatmasker/repeatmasker-4-0-5/RepeatMasker \
       -pa $NSLOTS \
       -lib $DATADIR/$REPLIB \
       -dir . \
       -xsmall \
       $DATADIR/$SEQFILE
rsync -a ./$SEQFILE.* $DATADIR/
```

1.5.3 qsub command

```
export DATADIR=`pwd`
qsub -v SEQFILE=seq.fa -v REPLIB=repeatLib.fa -pe smp 32 /path/to/repeatmasker.sh
```

1.5.4 useful one-liners

summarise genome size and repeat content across multiple results files in a folder (pattern matching specific to lepbase naming conventions):

```
perl -0777 -ne '$ARGV=~s/_-.+//;m/total.+?(\d+).+bases.+?(\d+)/s;print "$ARGV\t$1\t",($2/$1*100),"\n'
```

1.6 TRF

Tandem Repeats Finder [1] (link)

version 4.07b

1.6.1 Program parameters

```
$ /exports/software/trf/trf-4.07b/trf

Tandem Repeats Finder, Version 4.07b
Copyright (C) Dr. Gary Benson 1999-2012. All rights reserved.


Please use: /exports/software/trf/trf-4.07b/trf File Match Mismatch Delta PM PI Minscore MaxPeriod [c

Where: (all weights, penalties, and scores are positive)
  File = sequences input file
  Match  = matching weight
  Mismatch  = mismatching penalty
```

[1] Benson G (1999) Tandem repeats finder: a program to analyze DNA sequences *Nucleic Acids Research* **27**, 573-580.

```
  Delta = indel penalty
  PM = match probability (whole number)
  PI = indel probability (whole number)
  Minscore = minimum alignment score to report
  MaxPeriod = maximum period size to report
  [options] = one or more of the following
               -m    masked sequence file
               -f    flanking sequence
               -d    data file
               -h    suppress html output
               -r    no redundancy elimination
               -ngs  more compact .dat output on multisequence files, returns 0 on success. You may p
```

Recommended usage (plus html output suppression)

```
trf yoursequence.txt 2 7 7 80 10 50 500 -f -d -m -h
```

1.6.2 Program outputs

Outputs are written to input directory, using input filename suffixed by parameter values and file type

infile.2.7.7.80.10.50.500.mask

Hard-masked fasta format sequence file (option `-m`):

```
>seq1
GTTGTTTTTCTGAGACCGAAAAGGCTTGTTTTTATGTAGCAAGTTCAAGAACTGGTTTCG
GTATCGCAGGGTATGTATGGTCTGAGACGTATAACTGTCACGANNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNTATGATGAATGGGTAATTTTTAAAGTATCTCACATAACTTTTAGATG
TAGTAAGACCATGTTTGCGAATCGAATCTCTTACTGGTTGAACGCCAAAGGACCTTCCAT
...
>seq2
...
```

infile.2.7.7.80.10.50.500.dat

Text file with space delimited blocks (option `-d`):

```
Tandem Repeats Finder Program written by:

Gary Benson
Program in Bioinformatics
Boston University
Version 4.07b


Sequence: seq1



Parameters: 2 7 7 80 10 50 500


3404 3433 14 2.1 14 93 0 51 30 16 30 23 1.96 AGCAGCACTTGAGT AGCAGTACTTGAGTAGCAGCACTTGAGTAG
6344 6383 18 2.2 18 95 0 71 45 2 10 42 1.51 ATTATTATAAGGAATTAC ATTATTATAAGGAATTATATTATTATAAGGAATTACAT
```

```
...
6237875 6237915 16 2.6 16 96 3 75 0 4 48 46 1.23 TGTGTGTGTGTGCGTG TGTGTGTGTGTGCGTGTGTGTGTGTGTGCGTGTGT


Sequence: seq2



Parameters: 2 7 7 80 10 50 500


15030 15056 14 1.9 14 100 0 54 66 0 0 33 0.92 TTAAATAAATAAAT TTAAATAAATAAATTTAAATAAATAAA
...
```

Each block contains:

```
cols 1+2:  Indices of the repeat relative to the start of the sequence
col 3:     Period size of the repeat
col 4:     Number of copies aligned with the consensus pattern
col 5:     Size of consensus pattern (may differ slightly from the period size)
col 6:     Percent of matches between adjacent copies overall
col 7:     Percent of indels between adjacent copies overall
col 8:     Alignment score
cols 9-12: Percent composition for each of the four nucleotides
col 13:    Entropy measure based on percent composition
col 14:    Consensus sequence
col 15:    Repeat sequence
```

1.6.3 Grid engine wrapper

trf.sh:

```
#!/bin/bash

#$ -V
#$ -cwd
#$ -j y
#$ -o $JOB_ID.log

# define $DATADIR and $SEQFILE before/when calling this script, e.g.:
#       export DATADIR=`pwd`
#       qsub -v SEQFILE=seq.fa ...

WORKDIR=/scratch/$USER/trf/$JOB_ID
mkdir -p $WORKDIR
cp $DATADIR/$SEQFILE $WORKDIR
cd $WORKDIR
nice -n 10 /exports/software/trf/trf-4.07b/trf $SEQFILE 2 7 7 80 10 50 500 -f -d -m -h
rsync -a --remove-source-files ./${SEQFILE}* $DATADIR
```

1.6.4 qsub command

```
export DATADIR=`pwd`
qsub -v SEQFILE=seq.fa /path/to/trf.sh
```

1.6.5 parallel command

Split large sequence file into blocks with pyfasta, process each block using parallel to submit to gridengine with one slot per job, then combine outputs into single `.mask` and `.dat` files.

```
pip install pyfasta
pip install numpy
```

```
export DATADIR=`pwd`
export INFILE=seq # OMIT EXTENSION!
export SCRIPTDIR=~
pyfasta split -n 32 $INFILE
parallel -j 32 qsub -q nice.q -v SEQFILE={} -sync y $SCRIPTDIR/trf.sh ::: $INFILE.??.fa
cat $INFILE.??.fa.*.mask > $INFILE.fa.mask
cat $INFILE.??.fa.*.dat > $INFILE.fa.dat
```

```
rm *.log
rm $INFILE.??.fa.*.mask
rm $INFILE.??.fa.*.dat
rm $INFILE.??.fa
```

1.6.6 References

CHAPTER 2

Virtualisation

Contents:

2.1 Install Ubuntu VM

Initial KVM setup if this is the first VM:

```
virsh pool-define-as --name kvmpool --type dir --target /var/kvm/images
virsh pool-start kvmpool
```

Install a Ubuntu 14.04 VM direct from archive.ubuntu.com:

```
virt-install \
--name example \
--ram 4096 \
--disk path=/var/kvm/images/example.qcow2,size=24,format=qcow2 \
--vcpus=2,maxvcpus=6 \
--cpu host \
--os-type linux \
--os-variant ubuntutrusty \
--graphics none \
--console pty,target_type=serial \
--location 'http://uk.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' \
--extra-args 'console=ttyS0,115200n8 serial'
```

CHAPTER 3

Whole genome alignment

Contents:

3.1 Progressive Cactus

Progressive Cactus (link)

3.1.1 Typical Usage

```
/exports/software/progressiveCactus/bin/runProgressiveCactus.sh \
    /path/to/align.txt \
    /path/to/data/directory \
    /path/to/output/align.hal \
    --database kyoto_tycoon \
    --maxThreads 64
```

3.1.2 Input format

align.txt:

```
(tax1,(tax2,tax3));
tax1 /path/to/tax1.fa
tax2* /path/to/tax2.fa
tax3 /path/to/tax3.fa
```

Contains a newick tree on line 1 (don't forget the semicolon) followed by a set of mappings from taxon tames to (soft-masked) fasta files.

A taxon name with and asterisk is marked as reference quality (if no asterisks, all sequences will be treated as reference quality)

3.1.3 Gridengine wrapper

progressivecactus.sh (Progressive Cactus alignment followed by conversion to an Assembly Hub and a MAF file):

```
#!/bin/bash

#$ -V
#$ -cwd
#$ -j y
#$ -o $JOB_ID.log
#$ -pe smp 64

SCRATCH=/scratch/$USER/wga/$JOB_ID
mkdir -p $SCRATCH/$DIR
cd $SCRATCH

export PYTHONPATH=$PYTHONPATH:/exports/software/progressiveCactus/submodules/biopython
. /exports/software/progressiveCactus/environment

nice -n 10 /exports/software/progressiveCactus/bin/runProgressiveCactus.sh $DATA/$INFILE ./$DIR ./$DI

if ! [ -z $HUBFILE ]; then
nice -n 10 /exports/software/progressiveCactus/submodules/hal/bin/hal2assemblyHub.py --gcContent --ma
fi

if ! [ -z $MAFFILE ]; then
nice -n 10 /exports/software/progressiveCactus/submodules/hal/bin/hal2mafMP.py ./$DIR/$HALFILE $MAFFI
fi

rsync -av --remove-source-files ./$DIR $DATA/output

find ./$DIR -depth -type d -empty -delete
```

3.1.4 qsub command

```
qsub -v SEQFILE=seq.fa \
     -v DIR=input \
     -v DATA=/path/to/data \
     -v INFILE=align.txt \
     -v HALFILE=align.hal \
     -v HUBFILE=align.hub \
     -v MAFFILE=align.maf \
     -v REFGENOME=reference_genome_name \
     -pe smp 32 \
     /path/to/repeatmasker.sh
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search