1 CARP manual 1.1

bíogo written by: Dan Kortschak
Document organised by: Lu Zeng
December 14, 2017

krishna and igor are *ab initio* repeat family identification and annotation programs, that identify repeat element boundaries and family relationships from whole-genome sequence data. These programs build, refine and classify consensus models of putative interspersed repeats. krishna and igor are built using bíogo (https://github.com/biogo/biogo/), a bioinformatics library for the Go language.
Disclaimer This document is provided to assist researchers with linux command line experience. We have done our best to provide usable instructions, examples and advice, but users assume full responsibility for the output they generate and the authors accept no responsibility for user generated output from any programs or methods listed herein.

1.1 Prerequisites

1.1.1 Download Go

Available at (https://golang.org/dl).
For installation details, follow the instructions on the Go installation page.

1.1.2 Git

To perform the next step you will need Git to be installed. (Check that you have a git command before proceeding.)

If you do not have a working Git installation, follow the instructions on the Git download page.

1.1.3 Download bíogo Packages

Note: For convenience, add the workspace's bin subdirectory to your PATH, or add in $HOME/.profile.

* export PATH=$PATH:$(go env GOPATH)/bin

Download and install krishna and igor packages from github.

* go get -u github.com/biogo/examples/krishna
* go get github.com/biogo/examples/krishna/matrix

* go get github.com/biogo/examples/igor
* go get github.com/biogo/examples/igor/seqer
* go get github.com/biogo/examples/igor/gffer

1.1.4 Install CENSOR

Install censor to screen target genomes against a reference collection of repeats with masking symbols, as well as generating a report classifying all repeats found. CENSOR needs wu-blast/NCBI-blast and Bioperl installed.

CENSOR, along with instructions for installation, is available at CENSOR download page.

1.1.5 Install MUSCLE

Install MUSCLE to generate consensus sequences.
MUSCLE is available at MUSCLE download page. The installation information can be seen at MUSCLE Install page.

1.2 Example run

In this example, the human genome was downloaded as chromosomes (24 chromosomes) from UCSC into files called chr*.fa.

1.2.1 Use krishna to do pairwise alignment between human genome sequences

krishna-matrix helps you to align sequences by using a matrix table.

The default minimum hit length for krishna (-dplen) is 400bp, and minimum hit identity (-dpid) is 94%. The smaller the length and the lower the hit identity parameters you use, the more time and memory you will need.

If you want to change running parameters, for example, minimum hit length of 200bp, and minimum hit identity of 90%, just specify the parameters when running matrix -krishnaflags="-tmp=/scratch -threads=2 -log -filtid=0.9 -filtlen=200".

Now run the job:

* cd /data/rc003/lu/human (directory contains the sequences. Example path shown)
* matrix -krishnaflags="-tmp=./ -threads=2 -log -filtid=0.94 -filtlen=400" chr*.fa

-tmp: store the temporary files generated from krishna running, you can specify your own directory.
-threads: number of threads to use for alignment.
-filtid: minimum hit identity.
-filtlen: minimum hit length.

If genome sequence files are very big and consist of multiple contigs or scaffolds (>200MB), you can use bundle to split them into smaller files. For example,

* go get github.com/biogo/examples/bundle
* bundle -bundle 80000000 -in seq.fa

-bundle: specifies the total sequence length in a bundle. (default 20000000, 20MB).
-in: the genomes sequences you need to split.
Then run krishna job.

1.2.2 Use igor to report repeat feature family groupings in JSON format.

After running krishna, igor will take the pairwise alignment data to cluster repeat families.

* cat *.gff > hg_krishna.gff
* igor -in hg_krishna.gff -out hg94_krishna.json

1.2.3 Use seqer to generate consensus sequences from genome intervals .

seqer returns multiple fasta sequences corresponding to feature intervals described in the JSON output from igor.

gffer converts the JSON output of igor to gff. seqer will produce fastq consensus sequence output from either MUSCLE or MAFFT.

* gffer < hg94_krishna.json > hg94_krishna.igor.gff
* cat chr*.fa > hg19v37.mfa
* seqer -aligner=muscle -dir=consensus -fasta=true -maxFam=100 -subsample=true -minLen=0.95 -threads=12 -ref=hg19v37.mfa hg94_krishna.igor.gff

-fasta: Output consensus as fasta with quality case filtering
-maxFam: maxFam indicates maximum family size permitted (0 == no limit).
-minLen: Minimum proportion of longest family member.
-threads: Number of concurrent aligner instances to run.

1.3 Benchmarks

Genome	Krishna Threads	Genome DB Size	Krishna run time (hh:mm)	Igor run time (hh:mm)	Seqer run time (hh:mm)
Human	8	3.0G	~200	128:30	2:23
Bearded Dragon	8	1.8G	~23	73:11	<4
Anolis	6	1.8G	76:52	97:32	2:40
Chicken	4	1017M	5	<4	<1
Opossum	8	3.5G	~83	61:48	4:52
Platypus	8	2.0G	~99	191:34	10:16
Echidna	8	1.9G	< 360	12:43	9:02

* Analysis runs on a machine with 512GB RAM, running Red Hat Linux.

1.4 Repeat Library Annotation

Previous steps have generated repeat consensus sequences from the human genome, now we are going to annotate these repeat consensus sequences.
All the files used below can be found at
(https://data.mendeley.com/datasets/k88h5xnhcb/draft?a=d401233a-5af8-4879-81e8-c049b7133c8c).
All the codes used below can be found at
(https://github.com/luzengAdelaide/Biogo-document/tree/master/Codes)**.**

1.5 Annotate consensus sequences

Notes: All Java scripts used here need you to specify the directories where your input data is and where you want your output written.

1.5.1 Annotate consensus sequences with repeat families.

Use censor to annotate consensus sequences with the Repbase library. The Vertebrates.fa we use here is the Repbase vertebrates repeat libraries downloaded on 1st March, 2016.

* cd consensus
* cat *.fq > ConsensusSequences.fa
* censor -bprm cpus=8 -lib ~/Vertebrates.fa -lib ~/our_known_reps_20130520.fasta ConsensusSequences.fa

The censor output usually contains 5 files: ConsensusSequences.fa.map, ConsensusSequences.fa.aln, ConsensusSequences.fa.found, ConsensusSequences.fa.idx, ConsensusSequences.fa.masked.

1.5.2 Classify consensus sequences.

ConsensusSequences.fa and ConsensusSequences.fa.map are required in this step. You will also need to specify the directories for your input data and where you want your output written in the java code. Edit the source, compile and run.

* javac ClassifyConsensusSequences.java

* java ClassifyConsensusSequences

This should generate 5 output files: known.txt, partial.txt, check.txt, notKnown.fa, notKnown.fa.gff. Then we need to further annotate these notKnown.fa consensus sequences.

1.5.3 Filter potential protein sequences.

In this step, you will need to run the perl script reportsJ.pl (perl reportsJ.pl). This uses wu-blast/NCBI-blast blastx, you will also need the uniprot database. reportsJ.pl will ask for a list of databases to process, "libs.txt". This will include notknown.fa and notknown.fa.gff, put libs.txt into the current directory.

Notes: reportsJ.pl contains three parts: 1) identify uniprot, 2) GB_TE, 3) retroviruses. You can comment out two parts, and run each part separately in parallel to save time.
From these three steps, you will get three output files for following steps: 1) notKnown.fa.spwb.gff, 2) notKnown.fa.tewb.gff, 3) notKnown.fa.ervwb.gff

1.5.4 Get protein information from consensus sequences.

Another java program GetProteins.java will be used. It needs two input files: notKnown.fa, notKnown.fa.spwb.gff (Generated from previous step).

* javac GetProteins.java
* java GetProteins

You will get 2 output files: proteins.txt (a list of families that have been identified as proteins and the proteins they match); notKnownNotProtein.fa (a fasta file of the families that were not classified).

1.5.5 Check for simple sequence repeats (SSR).

Check for existence of SSR in the unknown sequences, using phobos. Phobos can be downloaded at (http://www.ruhr-uni-bochum.de/ecoevo/cm/cm_phobos.htm). We used executable: phobos-linux-gcc4.1.2 .

* phobos-linux-gcc4.1.2 -r 7 –outputFormat 0 –printRepeatSeqMode 0 notKnownNotProtein.fa > notKnownNotProtein.phobos

1.5.6 Identify the sequences that are SSRs from the phobos output.

phobos output will be used to identify SSRs: notKnownNotProtein.phobos.

* javac IdentifySSRs.java
* java IdentifySSRs

Your output will be a file called: SSR.txt

1.5.7 Generate annotated repeat library.

There are ten input files that are required to generate a repeat library:
1. ConsensusSequences.fa
2. ConsensusSequences.fa.map
3. notKnown.fa.tewb.gff
4. notKnown.fa.ervwb.gff

5. protein.txt
6. known.txt
7. GB_TE.21032016.fa
8. all_retrovirus.fasta
9. SSR.txt (if you do not have this, leave the definition in, it will generate error messages, but will not stop the program or affect the results.)
10. LA4v2-satellite.fa (you do not have this, or equivalent, as you didn't have any satellites, but leave the definition in-it will cause error messages, but will not stop the program or affect the results.)

* javac GenerateAnnotatedLibrary.java
* java GenerateAnnotatedLibrary

This will generate a library called "Human_Repeat_Library.fasta", you can change this to whatever you want to call your library.

1.6 Benchmarks2

Genome	Consensus sequences size	Censor first run time (hh:mm)	reportJ.pl (hh:mm)	phobos run time (hh:mm)
Human	38M	5:14	19:30	<00:10
Bearded Dragon	88M	22:21	<178	<00:10
New Bearded Dragon	88M	7:13	18:28	<00:10
Anolis	63M	9:42	78	<00:10
Chicken	18M	3:01	<24	<00:10
Opossum	60M	13:17	80	<01:00
Platypus	162M	17:34	115	<01:00
Echidna	59M	18:40	105	01:54

* Analysis run on a slurm machine with 4~16 cpus, and 8GB RAM, running Red Hat Linux.
** New Bearded dragon analysis used same bearded dragon genome, except it was run on a High Performance Computing machine with 32 cpus, running Red Hat Linux.

Finished! Good Luck!!!