CARP manual 1.1

b́iogo written by: Dan Kortschak
Document organised by: Lu Zeng
September 20, 2017

b́iogo (https://github.com/biogo/examples/) is a bioinformatics library for the Go language. b́iogo/krishna and b́iogo/igor are *ab initio* repeat family identifiction and annotation packages, that employ computational methods for identifying repeat element boundaries and family relationships from sequence data. b́iogo assists the runs of krishna and igor given a genome sequence and uses the output to build, refine and classify consensus models of putative interspersed repeats.

Prerequisites

Download Go

Available at (https://golang.org/dl). Developed and tested with version 1.5.2.
For installation details, follow the instructions on the Go installation page.

Git

To perform the next step you may want 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.

Download b́iogo Packages

Download krishna and igor packages from github.

* go get github.com/biogo/examples/krishna
* go get github.com/biogo/examples/igor
* go get github.com/biogo/external

Build b́iogo Packages

Usually, when you download a package, it will automatically build. However, some may not, so build packages from b́iogo to your home bin directory using the following command.

* go build -o $HOME/bin/seqer seqer.go

Install CENSOR

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

CENSOR is available at CENSOR download page.

Example run

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

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 these parameters, just add "-dplen=*bp" and "-dpid=0.*" in 'matrix.go' in the krishna source code: lines 102 and 132 (codes begin with "cmd := exec.Command").

Change the work directory that stores temporary files generated from krishna to your own directory (line 64 in matrix.go), then rebuild 'matrix.go'.

Notes: You need to rebuild the code every time after you change it.

Now run the job:

* cd /data/rc003/lu/human (directory contains the sequences)
* krishna-matrix chr-*.fa

If genome sequence files are very big and consist of multiple contigs or scaffolds (>200MB), split them into smaller files (using bundle.go).

* go build -o $HOME/work/bin/bundle bundle.go
* bundle -bundle 50000000 -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.

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

Use seqer to generate consensus sequences from genome intervals.

By default 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.

Benchmarks

Genome	Krishna Threads	Genome DB Size (G)	Krishna run time (hh:mm)	Igor run time (hh:mm)	Seqer run time (hh:mm)
Human	8	3.0	~200	128:30	2:23
Bearded Dragon	8	1.8	~23	73:11	<4
Anolis	6	1.8	76:52	97:32	2:40
Chicken	4	1.0	5	<4	<1
Opossum	8	3.5	~83	61:48	4:52
Platypus	8	2.0	~99	191:34	10:16
Echidna	8	1.9	< 360	12:43	9:02

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

Repeat Library Annotation

Previous steps have generated repeat consensus sequences from the human genome, now we are going to annotate these repeat consensus sequences.

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. The java scripts are available on github (https://github.com/luzengAdelaide/TE-Denovo-Annotation/JavaScripts)**.**

Annotate consensus sequences with repeat families.

Use censor to annotate consensus sequences with the Repbase library.

* cd consensus
* cat *.fq > ConsensusSequences.fa
* censor -bprm cpus=8 -lib ~/Verterbrates.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 and ConsensusSequences.fa.masked.

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.

Filter potential protein sequences.

reportsJ.pl (perl reportsJ.pl) identifies consensus sequences that correspond to 1) proteins from uniprot, 2) reverse transcriptase and TE sequences from NCBI (GB_TE), and 3) retrovirus. This uses wu-blast blastx, you will also need the uniprot, GB_TE and retrovirus database. reportsJ.pl will ask for a list of databases to process, "libs.txt". This process will include notknown.fa and notknown.fa.gff, put libs.txt into the current directory.

Notes: GB_TE database can be downloaded using our perl code (GB_TE_efecth.pl), (https://github.com/luzengAdelaide/TE-Denovo-Annotation/PerlScripts)**.**

At the end of this filtering step you will have three output files: 1) notKnown.fa.spwb.gff, 2) notKnown.fa.tewb.gff, 3) notKnown.fa.ervwb.gff

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).

Check for simple sequence repeats (SSR).

Check for existence of SSR in the unknown sequences, using phobos. Phobos can be downloaded from Phobos download page.

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

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

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.

Benchmarks2

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