RGCAT - reference-based genome comparison and annotation tool

This program is a reference-based genome comparison and annotation tool to carry out common tasks for genome comparison, pan-genome construction, and annotation transformation.

With release of high-quality genome assemblies and emergence of pan-genomes, there is an increasing need to build pan-genome, create chain file between two assemblies and transform annotation files between them. To handle these tasks, we developed RGCAT, an efficient tool for construction of pan-genome with assemblies added one by one, generation of chain file, and liftover of annotation file in vcf or gff format, based on the alignment of assemblies. To improve accuracy, abnormal alignment is removed firstly according to alignment quality and overlap between query-to-target and target-to-query alignment.

It is a versatile tool to carry out common tasks for comparing genome, building pan-genome, and tranfering annotation. Typical tasks include: (1) genome alignment between two assemblies based on minimap2; (2) alignment filter based on query-to-target and target-to-query mapping result in PAF format; (3) reference-based pan-genome construction with other assemblies added one by one; (4) chain and vcf file generation based on query-to-target and target-to-query mapping result in PAF format; (5) annotation liftover based on chain file and annotation file in vcf, gff/gtf or bed format.

Installation

In default, RGCAT need these prerequisites:

- Perl (v5.28.1, maybe lower or higher)
- minimap2 (2.26-r1175)

This program doesn't need install. You can run it using absolute path. It has been tested on Linux.

Usage

Run it without any parameter `perl RGCAT.pl` , it will print usage on the screen. You can also use help command `perl RGCAT.pl --help|-h` to show usage message.

This program calls minimap2 to do alignment between query and target/reference assemblies. To filter misalignment, the program does both of query-to-target and target-to-query alignments, filters secondary alignments, and only keeps alignments overlapped in both of query-to-target and target-to-query alignments. Moreover, the query sequence file, query gff file, and contig prefix of query should be in the same order. In default, The program will create a subdir inside outdir and generate chain, vcf and bed files for both of query-to-target and target-to-query alignments. To achieve a better vcf transfromation in liftvcf, we used chain and vcf files for both of query-to-target and target-to-query alignments.

1. Commands

   > --command|-c

   This program need to assign task by command parameter. The details about each task was explained bellow.
   Use parameter `--command gencmp` to compare between query and target/reference genome assemblies.
   Use parameter `--command pangen` to construct pan-genome based on reference genome assembly and other query genome assemblies.
   Use parameter `--command liftgff` to lift gff file based on chain file.
   Use parameter `--command liftvcf` to lift vcf file based on chain file.
2. Input files

This program needs several necessary input files for all tasks.

> --refseq|-rs

Use parameter `--refseq referenece.fasta` to assign a reference/target sequence file in fasta format.

> --queseq|-qs

Use parameter `--queseq query.fasta` to assign a query sequence file in fasta format or `--queseq query_sequence_list_file` to assign a file for all query sequence files (each line is a name of query sequence file).

> --outdir|-o

Use parameter `--outdir out_directory` to assign output directory.

The task `--command liftgff` need genome annotation file in GFF format.

> --ingff|-ig

Use parameter `--ingff input.gff` to assign GFF file which needs to be transformed.

The task `--command liftvcf` need variant file in VCF format.

> --invcf|-iv

Use parameter `--invcf input.vcf` to assign VCF file which needs to be transformed.

The task `--command gencmp` or `--command pangen` can provide genome annotation file for annotation comparison using the following parameters.

> --refgff|-rf

Use parameter `--refgff referenece.gff` to assign a reference/target GFF file.

> --quegff|-qf

Use parameter `--quegff query.gff` to assign a query GFF file or `--quegff query_gff_list_file` to assign a file for all query GFF files (each line is a name of query GFF file).

3. Output files

This program produces several output files in regular format.

- *.paf (PAF format, genome comparison file, https://github.com/lh3/miniasm/blob/master/PAF.md)
- *.fasta (FASTA format, sequence file, https://seq2fun.dcmb.med.umich.edu/FASTA/)
- *.bed (BED format, bed file, http://genome.ucsc.edu/FAQ/FAQformat.html#format1)
- *.chain (CHAIN format, chain file, http://genome.ucsc.edu/goldenPath/help/chain.html)
- *.vcf (VCF format, variant file, http://genome.ucsc.edu/goldenPath/help/vcf.html)
- *.gff (VCF format, annotation file, http://genome.ucsc.edu/FAQ/FAQformat.html#format3)

4. Detail parameters

```
Usage: perl RGCAT.pl --command|-c <command> [options]

The following options are necessary.
--command|-c <command>: a command for different tasks.
        The details about each task was explained bellow.
        gencmp      compare between query and target/reference genome assemblies
        pangen      construct pan-genome based on reference genome assembly and other query genome assemblies
        liftgff     lift gff file based on chain file
        liftvcf     lift vcf file based on chain file
--refseq|-rs <*>:       a reference/target sequence file in fasta format.
--queseq|-qs <*>:       a query sequence file in fasta format or a file for all query sequence files (each line is a na
--outdir|-o <*>:        output directory.

The following options are optional.
--refgff|-rf <*>:       a reference/target gff file. It can be used to extract genes in referenece/target unique sequen
--quegff|-qf <*>:       a query gff file or a file for all query gff files (each line is a name of query gff file). It
--cigarpreset|-cx <*>:  asm5/asm10/asm20, asm-to-ref mapping, for ~0.1/1/5% sequence divergence (minimap2 option).
--secondfilter|-sf <*>: filter secondary alignments ("y" or "n", "y" in default).
--overfilter|-of <*>:   filter alignments unsatisfied overlap coverage rate (assigned by --aligncov) between query-to-t
--aligncov|-ac <*>:     alignment coverage rate between query-to-target and target-to-query (0.8 in default).
--minuniqlen|-ul <*>:   minimum length of unique sequence in assemblies (1000 in default).
--maxuniqide|-ui <*>:   maximum identity of unique sequence in assemblies (0.9 in default).
--maxuniqcov|-uc <*>:   maximum coverage of unique sequence in assemblies (0.8 in default).
--genuniqcov|-gc <*>:   minimum gene coverage rate of unique sequence in assemblies (0.5 in default).
--chain|-cn <*>:        create chain file between query and target/reference genome assemblies ("y" or "n", "y" in defa
--vcf|-vc <*>:          create vcf file between query and target/reference genome assemblies ("y" or "n", "y" in defaul
--bed|-b <*>:           create bed file between query and target/reference genome assemblies ("y" or "n", "y" in defaul
--version|-v:           print the usage information.
--help|-h:              print the usage information.

These parameters belong to command pangen.
--ctgpre|-cp <*>:       contig prefix for query sequence. It could be a contig prefix for a query or a file for all con

These parameters belong to command liftvcf.
--invcf|-iv <*>:        input vcf. It could be coodinated in query or reference/target sequences.
--outvcf|-ov <*>:       output vcf. The converted result. If input vcf is coodinated in query, output vcf should be coo
--failvcf|-fv <*>:      fail vcf. The unconverted result.
--transform|-tf <*>:    transform type ("query-to-target" or "target-to-query").

These parameters belong to command liftgff.
--ingff|-ig <*>:        input gff. It could be coodinated in query or reference/target sequences.
--outgff|-og <*>:       output gff. The converted result. If input gff is coodinated in query, output gff should be coo
--failgff|-fg <*>:      fail gff. The unconverted result.
--transform|-tf <*>:    transform type ("query-to-target" or "target-to-query").
```

Examples

1. For gencmp command, this program takes two assemblies as input and produces genome comparison results, which include genome alignment in PAF format, genome comparison in CHAIN format, genome coodinate comparison in BED format, and genome sequence variants in VCF format.
   Example commands:

   > `perl RGCAT_v1.pl -c gencmp -rs <reference.fa> -qs <query.fa> -o <output_dir>`

   > `perl RGCAT_v1.pl -c gencmp -rs <reference.fa> -qs <query.fa> -rf <reference.gff> -qf <query.gff> -o <output_dir>`

2. For pangen command, this program takes one reference assembly and several query assemblies as input and produces pan-genome gradually with query genome added one by one. The final pan-genome sequence in FASTA format and annotation in GFF format will be created in the last produced subdiretory in side output directory.
   Example commands:

   > `perl RGCAT_v1.pl -c pangen -rs <reference.fa> -qs <listfile_of_query.fa> -cp <list_file_of_contig_prefix> -o <output_dir>`

   > `perl RGCAT_v1.pl -c pangen -rs <reference.fa> -qs <listfile_of_query.fa> -rf <reference.gff> -qf <query.gff> -cp <list_file_of_contig_prefix`

3. For vcf transformation, this program will produce tranformed output vcf and failed vcf files. Important: vcf file should be sorted before running liftvcf.

Example command:

> ```
> perl RGCAT_v1.pl -c liftvcf -rs <reference.fa> -qs <query.fa> -o <output_dir> --invcf <input.vcf> --outvcf <output.vcf> --failvcf <fail.vcf>
> ```

4. For gff transformation, this program will produce tranformed output gff and failed gff files.

   Example command:

   > ```
   > perl RGCAT_v1.pl -c liftgff -rs <reference.fa> -qs <query.fa> -o <output_dir> --ingff <input.gff> --outgff <output.gff> --failgff <fail.gff>
   > ```

Contact

If you have any question or suggestion, please contact Wanfei Liu. Email: liuwanfei@caas.cn