PlantPseudo’s manual for singularity container users

Overview

Pseudogenes are important resources in understanding the evolutionary history of genes and genomes. This pseudogene detection pipeline was used for pseudogene identification in plant species.

The pipeline is executed using command line options on Linux systems. The pipeline has now packaged into a singularity container which would be easier for readers to use and reproduce the results.

All code is copiable, distributable, modifiable, and usable without any restrictions.

Installation

On Linux systems, the singularity recipe file can be downloaded through git command (git clone https://github.com/bjfupoplar/PlantPseudo.git). The singularity should be installed before the installation (http://singularity.lbl.gov/install-linux#installation-from-source; it is required to be run as root to get a properly installed Singularity implementation; the stable version is 2.5.2). We have developed the container based on singularity version 2.5.2, and the container built with higher version may not work normally in lower version.

Then in the PlantPseudo you will find a recipe file named Singularity, simply put it into a directory and run (running it may require root privilege):

```
## Build a CentOS image using Singularity
$ sudo singularity build PlantPseudo.img Singularity
```

You can see the directory PlantPseudo.img in the container using the command `ls`.

We also provide a singularity image which can be downloaded through wget (wget ftp://106.2.11.172/pub/paper/Singularity/*; the size is ~300 Mb).

Usage

# Input data

-- rawFa: contains a file which is the unmaksed genome for each species.

```
>Chr01
AAACCCTAAACCCTAAACCCTAAACCCTAAACCGTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTA
```

-- repeatMaskedFa: contains a file which is entire repeat masked genome dna sequence from that species in FASTA format.

```
>Chr01
AAACCCTAAACCCTAAACCCTAAANNNNNNNNNNNNNNNNNNNNNNNNNACCCTAAACCCTAAACCCTAAACCCTA
```

-- pep: contains a FASTA file for all the proteins in the species.

```
>Potri.002G048200
MNPYLTVKQEYAGSSLLPLSGGDEPPTMMLPPQPMEGLHDTGPPPFLTKTFDMVDDPMTNHIVSWSRGGFSFVVWDP
```

-- gff: The GFF (General Feature Format) format consists of one line per feature, each containing 9 columns of data, plus optional track definition lines.

```
Chr02   phytozomev1   gene   4173   8240   ID=Potri.002G000100.v3.0;Name=Potri.002G000100
```

-- lncrna: if provided, the pipeline will detect the associations between lncRNAs and Pseudogenes/Genes.

```
Chr02   80114   80235   TCONS_00078309   +
```

-- repeatMaskedGff: if provided, the pipeline will identifty helitron-associated pseudogenes. (The file is the output of RepeatMasker, which is a gff3 format)

```
SW    perc   perc   perc   query      position in query    matching    repeat    position in repeat
score   div. del. ins.   sequence   begin end    (left)   repeat   class/family   begin    end    (left)   ID
46    2.0    2.0    0.0    Chr01    2    51 (50495340) + (CCCAAAC)n    Simple_repeat    1    51    (0)    1
```

# How to run:

To reproduce the result of the seven plant species, you can run the commond:

```
$ sudo singularity exec PlantPseudo.img git clone https://github.com/bjfupoplar/PlantPseudo.git
$ sudo singularity exec PlantPseudo.img sh /root/PlantPseudo/workflow.sh
```

Through these two commands, a folder named PlantPseudo will be created in the root's home: /root. The script provided above invokes wget to download all the sample data and the genome data from external repositories (wget ftp://106.2.11.172/pub/paper/sample.data.tar.gz; wget ftp://106.2.11.172/pub/paper/genome.tar.gz). Two folders named sample.data and data will be created. When finished (it may takes several days depends on the genome size and gene numbers), you will see the input data and the results for the sample data and each plant species.

# Output:

-- result1: Exonerate alignment result

```
Command line: [PlantPseudo/software/exonerate-2.2.0-x86_64/
Hostname: [forestry]

C4 Alignment:
------------
    Query: Potri.002G049000
    Target: Chr02
    Model: protein2genome:local
  Raw score: 122
  Query range: 125 -> 175
  Target range: 3201791 -> 3201945

  126 : LeuGluSerAlaIleLeuThrThrValValValValSerLeuThrMetTyrThrPh :    144
    |||.!.|||||||||:!!|||   !!.!!.!|||||!!!||||||:!!|||||||||
    LeuHisSerAlaIleIleThrPheAlaAlaValValCysLeuThrLeuTyrThrPh
 3201792 : TTACATTCTGCCATCATAACTTTTGCGGCCGTGGTTTGTCTCACTCTGTACACTTT : 3201846

  145 : eTrpAlaAlaArgArg----GlyHisAspPheAsnPheLeuGlyProPheLeuPhe :    161
    |! !||||||||||||####||||||||||||!:!||||||||||||||||||! !
    e***AlaAlaArgArg####GlyHisAspPheSerPheLeuGlyProPheLeuSer
 3201847 : CTAGGCGGCAAGGAGACTGAGGTCATGATTTCAGCTTCCTTGGGCCCTTCTTGTCT : 3201901

  162 : GlyAlaValMetValLeuMetValPheAlaPheIleGlnIle :    175
    !.!:!!:!!!!:!.!:!!:!!:!!||| !!.!.|||!:!:!!
    AlaSerLeuIleAlaIleLeuLeuPheProLeuIleArgVal
 3201902 : GCTTCCCTGATTGCTATTCTGCTGTTTCCTCTGATCCGGGTA : 3201945

vulgar: Potri.002G049000 125 175 . Chr02 3201791 3201945 + 122 M 24 72 F 0 4 M 26 78
```

-- result2: The pseudogene set

```
pgId   pgChr   pgStart  pgEnd   pgStrand   pgpolyA  expect   ident   stop1   stop2   fShift1  fShift2  numofIntrons
intronPos   paln   pId   pChr   pStart   pEnd   pStrand  Frac   DupType
Chr02|2960109-2960404   Chr02   2960109  2960404  -   1   1.4e-22  48.485   1   0   1   0   0   ---   99.0
Potri.008G159700 Chr08   10836144   10841778   +   0.16722972973   WGDDUP
```

-- result3: The classification results of lncRNAs whether it is closer to pseudogenes or genes

```
type    distance    lncRChr    lncRstart    lncRend Chr    start    end
genedist 311    Chr02    257369    257751    TCONS_00087176    Chr02    257680    257051    Potri.002G004200
```

- result4: The Classification results of lncRNAs which closer to genes; promorter: Proximal upstream region associated nonTE lncRNA loci (The opposite transcription direction); body: Gene body associated nonTE lncRNA loci; Proximal upstream region associated nonTE lncRNA loci (The same transcription direction); f: Tail to tail; distant: Distant nonTE lncRNA loci (>2 kb)

```
promoter      4
body     27
Co    11
f     0
distant 34
```

-- result5: The detailed Classification results of lncRNAs which closer to genes

```
Classification     Type     distance     lncRChr   lncRstart     lncRend   lnRNA     Chromosome     Start     End
Gene/Pseudogene
Body  associated  genedist   311    Chr02    257369    257751    TCONS_00087176    Chr02    257680    257051
Potri.002G004200
```

- result6: The Classification results of lncRNAs which closer to pseudogenes; promorter: Proximal upstream region associated nonTE lncRNA loci (The opposite transcription direction); body: Gene body associated nonTE lncRNA loci; Proximal upstream region associated nonTE lncRNA loci (The same transcription direction); f: Tail to tail; distant: Distant nonTE lncRNA loci (>2 kb)

```
promoter      0
body     1
Co    0
f     0
distant 0
```

-- result7: The detailed Classification results of lncRNAs which closer to pseudogenes.

```
Classification    Type    distance    lncRChr  lncRstart    lncRend    lnRNA    Chromosome    Start    End
Gene/Pseudogene
Body  associated  pgdist   539   Chr02   4727923  4728209  TCONS_00087235   Chr02   4728462  4727576
Chr02|4727576-4728462
```

For example:

Folder: /root/PlantPseudo/data/1.Populus

Input files	
-- pep	pt.pep
-- gff	pt.genome.gff3
-- rawFa	pt.raw.fa
-- repeatMaskedFa	pt.repmasked.fa
-- lncrna	lncrna.gff
Output files	
-- result1	exonerate.out
-- result2	final.pg.xls
-- result3	compare.xls
-- result4	Pg.Pseudo.distance.xls
-- result5	Pg.Classfication.xls
-- result6	Gene.Pseudo.distance.xls
-- result7	Gene.Classifcation.xls

# Run with your own data:

Provide your own genomic data using the parameters below:

```
$ sudo cd /root/PlantPseudo/PlantPseudo
$ sudo mkdir result
$ sudo mkdir own.data
# Put your own data into the own.data directory

$ sudo singularity shell PlantPseudo.img
$ sudo cd PlantPseudo
$ sudo cd bin
$ sudo perl pipeline.pl --scriptDir ../script –gff ../own.data/genome.gff3 --pep ../own.data/sample.pep
--rawFa ../own.data/raw.fa  --lncrna lncrna.gff --repeatMaskedFa ../own.data/repmasked.fa --eValueE 5
--idenThresh 20 --lenThresh 30 --proThresh 0.05 --qs 1 --mLenPse 50 --mLenIntron 50 --dirfile pathfile.txt
--outDir ../result
```

Workflow description

1. step1

- script: Gff2Genepos.py

- description: Extract gene position information from gff3 file

- output table: Chromosome start end strand gene

2. step2

- script: fa-mask.py

- description: masked genic regions

- output: Repeatmasked- and genic-Masked genome sequence

3. step3

- script: exonerate

- description: align the protein sequences to the masked genome

- output table：Chromosome programe gene_partion start end length strand . gene

4. step4

- script: ExtractExonerateOut.py

- description: extract the best alignment result

- output table: Query id Subject id % identity alignment length mismatches gap openings q. start q. end s. start s. end e-value bit score

5. step5

- script: ParseBlast.py

- description: Filter the alignment result using parameter -E Evalue -I (identity) -L (match length) -P (length) -Q 1 (protein or subject for depth)

- output table: Query id Subject id % identity alignment length mismatches gap openings q. start q. end s. start s. end e-value bit score

6. step6

- script: Pseudo_step1.py

- description: Consolidate multiple matches between the same intergenic seq-query protein pairs.

- output table: Chromosome [genome:start,en] [protein;start,end] [E value] strand gene

7. step7

- script: Pseudo_step2.py

- description: Combine matches with different proteins at once to construct pseudoexons.

- output table: Chromosome gene [genome:start,end] [protein;start,end]

8. step8

- script: Pseudo_step3.py

- description: get the coordinates of pseudogenes on the subject sequences

- output table: output table: Gene Chromosome|start-end

9. step9

- script: FastaManager.py

- description: Extract Pseudoexon regions

- output: Pseudoexon sequences

10. step10

- script: BlastUtilityv2.py

- description: Perform realignment using tfasty software

- output: tfasty output

11. step11

- script: Pseudo_step4.py

- description: Extract tfasty output infromation

- output:

- Gene Chromosome|start-end

- Gene_length Genome_subject_length identity% E_value Smith-Waterman_score Smith-Waterman_%identity Smith-Waterman_simlarity alignment_start_end

- seq1 (Protein sequences)

- seq2 (Genome sequence)

12. step12

- script: CheckStrand.py

- description: Check the alignment orientation

- output table: Chromosome start end strand pseudogene

13. step13

- script: PolyACheck.py

- description: Check if there are any PolyA signal in the downsteam of pseudogene

- output table: Chromosome start end strand pseudogene maxCount maxPos maxStr signalPos kind

14. step14

- script: CheckIntron.py

- description: Extract intron information from exonerate

- output table: exonerate output

15. step15

- script: SumTablev2.py

- description: Combine the previous outputs

- output table: pgId pgChr pgStart pgEnd pgStrand pgpolyA expect ident stop1 stop2 fShift1 fShift2 numofIntrons paln pId

16. step16

- script: GetIntronfracv2.py

- description: Calculate the match length ratio against the full length protein length

- output table: pgId pgChr pgStart pgEnd pgStrand pgpolyA expect ident stop1 stop2 fShift1 fShift2 numofIntrons intronPos paln pId pChr pStart pEnd pStrand Frac

17. step17

- script: PgClassification.py
- description: Filter the pseudogene output (The match length ratio <0.05 and the pseudogene length<30 were removed)
- output table: pgId pgChr pgStart pgEnd pgStrand pgpolyA expect ident stop1 stop2 fShift1 fShift2 numofIntrons intronPos paln pId pChr pStart pEnd pStrand Frac

18. step18

- script: Pggff.py,mcscanformatv2.py,Mcscan2Pglstv2.py
- description: Prepare for the input for MCscanX.
- output: WGD-derived pseudogene list is generated.

19. step19

- software: MCScanX
- description: The WGD-derived pseudogenes were detected using MCScanX.
- output: MCScanX output.

20. step20

- script: FinalPglst.py

- description: The type of pseudogene is added to the last column.

- output table: pgId pgChr pgStart pgEnd pgStrand pgpolyA expect ident stop1 stop2 fShift1 fShift2 numofIntrons intronPos paln pId pChr pStart pEnd pStrand Frac DupType

21. step21

- script: DistanceComparev5.1.py

- description: The distance between Genes/Pseudogenes and lncRNAs

- output table: type distance lncRChr lncRstart lncRend Chr start end