Created time: Monday, September 03, 2012
Version: V1.30
Email: thinkhighly@163.com
Authors: Jinzhuang Dou, Xiaoteng Fu, Xiaoyu Mu, Shi Wang
Key Laboratory of Marine Genetics and Breeding, College of Marine Life Sciences, Ocean University of China, 5 Yushan Road, Qingdao 266003, China

Overview of RADtyping pipeline

RADtyping is a user-defined perl procedure for performing *de novo* RAD genotyping in mapping populations. It has three major features:

- **Codominant and dominant genotyping**

  RAD sequencing can generate two types of markers (i.e. codominant or dominant), which differ by whether a SNP disrupts the recognition site or not. Most RAD studies only score codominant markers, while for dominant markers, they are still largely unexplored. RADtyping enables both codominant and dominant genotyping, thus maximizing the genotypic information for a given amount of sequencing.

- **High level of genotyping accuracy**

  Reconstructing reference sites correctly from sequencing data is critical for accurate genotyping. RADtyping enables setting up high-quality representative reference sites by removing repetitive sites and sequencing errors efficiently, which, in combination with our optimized genotyping algorithms, contributes to a high level of genotyping accuracy.

- **Convenience**

  For biologists who are not familiar with Linux or Perl programming, RADtyping pipeline allows you to finish the whole analysis by typing just one command line (under default parameters). In addition, RADtyping allows transforming genotyping results into a Joinmap format for construction of genetic maps.

Before you beigin

The following software need to be installed. Note, their most recent versions have not been tested.
Stacks (version 0.99997)[1] *SOAP2 (version 2.1.1b)*[2]
Velvet(version 1.1)[3]
Make sure that the Perl scripts listed below are available in your current working path, so you can call them directly from a single command line:
name_change.pl
reads_proc.pl
ref_build.pl
ref_filter.pl

reads_filter.pl
reads_map.pl
codom_calling.pl
dom_calling.pl
joinmap_trans.pl
RADtyping.pl

◆1 Reads preprocessing

Description:
Raw reads are first preprocessed to remove unreliable ones with no restriction site, ambiguous basecalls (N), long homopolymer regions or excessive low-quality positions. The obtained high-quality reads are stored in the directory "proc_data" and are ready for subsequent analysis.

Usage: reads_proc.pl [option]

Parameter	Format	Description
-i	<str>	input directory containing progeny data
-p1	<str>	input file of parent 1
-p2	<str>	input file of parent 2
-b	<str>	target restriction site for BsaXI: [ATGC]{9}AC[ATGC]{5}CTCC[ATGC]{7}
-f	<int>	f=1 palindromic site; f=2 non-palindromic site [2]
-l	<int>	read length
-Q1	<int>	threshold for low quality score [20]
-Q2	<int>	maximum no. of low-quality bases [10]
-S	<float>	discard reads with homopolymers > (S × l) [0.3]
-h		display the help information

◆2 Reference sites reconstruction

Description:
All preprocessed reads from two parents are combined and assembled into exactly matching read clusters (i.e. representing individual alleles), and then "allele" clusters are further merged into "locus" clusters by allowing certain mismatches. A collection of consensus sequences from all "locus" clusters comprises the representative reference sites. These sites are further classified into parent-shared and parent-specific sites for subsequent codominant and dominant genotyping. Reference sites are stored in two output files, i.e. "ref/ref_codom" and "ref/ref_dom".

Usage: ref_build.pl [option]

Parameter	Format	Description
-p1	<str>	input file of parent 1
-p2	<str>	input file of parent 2
-m	<int>	min. depth required to create an "allele" cluster [3]

-M	<int>	max. mismatches allowed for merging “allele” clusters [2]
-p	<int>	enable parallel execution with multiple threads [14]
-pe	<str>	pe=”s” single-end sequence, pe=”p” paired-end sequence[s]
-h		display the help information

◆3 Obtaining high-quality reference sites

Description:

To obtain high-quality (HQ) reference sites for reliable genotyping, reference sites are filtered by excluding those that are either not supported by parental reads in sufficient depth or derived from repetitive genomics regions. Three methods are available, i.e. mixed Poisson model, mixed normal model and threshold method. The mixed Poisson and normal models are recommended if your data fit one of them, which outperform the classic threshold method in removal of repetitive sites[4]. HQ reference sites are stored in two output files, i.e. “ref/HQ_ref_codom” and “ref/HQ_ref_dom”.

Usage: ref_filter.pl [option]

Parameter	Format	Description
-m	<str>	choosing a filtering method [O] P: mixed Poisson model; N: mixed normal model; T: threshold method; O:choose the optimal model automatically.
-h		display the help information

◆4 Reads mapping

Description:

The preprocessed reads from two parents and their progenies are separately mapped to the obtained HQ reference sites. All mapping results are stored in a directory named “reads_mapping”.

Usage: reads_map.pl [option]

Parameter	Format	Description
-M	<int>	choosing a match mode [4] 0: exact match; 1: one mismatch allowed; 2: two mismatches allowed; 4: find the best hit.
-l	<int>	read length
-h		display the help information

◆5 Codominant genotyping

Description:

For each locus, genotype is determined using the maximum likelihood method (ML)[5]. The most likely genotype is assigned based on a likelihood ratio test (LRT) between two possible genotypes (i.e. homozygote and heterozygote). Codominant genotypes are stored in two output files, i.e. “genotype/all_codom” and “genotype/poly_codom”.

Usage: codom_calling.pl [option]

Parameter	Format	Description
-a	<float>	significance level of LRT test [0.05]
-p	<float>	least percentage of genotyped progenies [0.8]
-o1	<str>	output file of all codominant markers [genotype/all_codom]
-o2	<str>	output file of polymorphic codominant markers [genotype/poly_codom]
-h		display the help information

◆6 Dominant genotyping

Description:

Dominant markers are scored as “presence” or “absence” to reflect whether the recognition site is intact or disrupted. Dominant genotyping is performed by evaluating the reliability of observed tag presence or absence (see Methods for algorithm details). Dominant genotypes are stored in two output files, i.e. “genotype/all_dom” and “genotype/poly_dom”.

Usage: dom_calling.pl [option]

Parameter	Format	Description
-p	<str >	least percentage of genotyped progenies [0.8]
-o1	<str>	output file of all dominant genotypes [genotype/all_dom]
-o2	<str>	output file of polymorphic dominant markers [genotype/poly_dom]
-h		display the help information

◆7 Format transformation for Joinmap

Description:

To facilitate downstream linkage mapping using Joinmap program, this step reorganizes the obtained genotyping results in a format ready for Joinmap analysis.

Usage: joinmap_trans.pl [option]

Parameter	Format	Description
-c	<str>	input file of codominant genotypes [genotype/poly_codom]
-d	<str>	input file of dominant genotypes [genotype/poly_dom]
-c1	<str>	output Joinmap file of codominant genotypes [genotype/codom_JM]

-d1	<str>	output Joinmap file of dominant genotypes [genotype/dom_JM]
-h		display help information

◆8 The integrated pipeline – RADtyping

Description:

The perl script **RADtyping.pl** allows you to finish the whole analysis by using just one command line (under default parameters).

Usage: RADtyping.pl [option]

Parameter	Format	Description
-c	<str>	input directory containing progeny data
-p1	<str>	input file of parent 1
-p1	<str>	input file of parent 2
-pe	<str>	pe="s" single-end sequence, pe="p" paired-end sequence[s]
-l	<int>	read length
-h		display help information

Example Usage

If all your input files are stored in a directory called "rawdata", which contains Illumina sequencing data for two parents and twenty progenies:

```
mgb@M:~/djz/rawdata$ ls
Progeny_001.fastq  Progeny_007.fastq  Progeny_013.fastq  Progeny_019.fastq
Progeny_002.fastq  Progeny_008.fastq  Progeny_014.fastq  Progeny_020.fastq
Progeny_003.fastq  Progeny_009.fastq  Progeny_015.fastq  Parent_001.fastq
Progeny_004.fastq  Progeny_010.fastq  Progeny_016.fastq  Parent_002.fastq
Progeny_005.fastq  Progeny_011.fastq  Progeny_017.fastq
Progeny_006.fastq  Progeny_012.fastq  Progeny_018.fastq
```

The high-quality (HQ) reads can be obtained by running:

```
mgb@M:~/djz/$ perl reads_proc.pl    -i rawdata -p1 rawdata/Parent_001.fastq -p2
rawdata/Parent_002.fastq -b [ATGC]{9}AC[ATGC]{5}CTCC[ATGC]{7} -l 27
```

The obtained HQ reads are stored in a directory named "proc_data".

```
mgb@M:~/djz/proc_data$ ls
Progeny_001.fasta  Progeny_007.fasta  Progeny_013.fasta  Progeny_019.fasta
Progeny_002.fasta  Progeny_008.fasta  Progeny_014.fasta  Progeny_020.fasta
Progeny_003.fasta  Progeny_009.fasta  Progeny_015.fasta  P1.fasta
Progeny_004.fasta  Progeny_010.fasta  Progeny_016.fasta  P2.fasta
Progeny_005.fasta  Progeny_011.fasta  Progeny_017.fasta
Progeny_006.fasta  Progeny_012.fasta  Progeny_018.fasta
```

If your data are pair-end, it is necessary to pair the data together in the directory "proc_data".

```
mgb@M:~/djz/proc_data$ ls
```

```
Progeny_001_p.fasta    Progeny_007_p.fasta    Progeny_013_p.fasta
Progeny_019_p.fasta    Progeny_002_p.fasta    Progeny_008_p.fasta
Progeny_014_p.fasta    Progeny_020_p.fasta    Progeny_003_p.fasta
Progeny_009_p.fasta    Progeny_015_p.fasta    Progeny_004_p.fasta
Progeny_010_p.fasta    Progeny_016_p.fasta    Progeny_005_p.fasta
Progeny_011_p.fasta    Progeny_017_p.fasta    Progeny_006_p.fasta
Progeny_012_p.fasta    Progeny_018_p.fasta    P1_p.fasta    P2_p.fasta
Progeny_001.fasta      Progeny_007.fasta      Progeny_013.fasta    Progeny_019.fasta
Progeny_002.fasta      Progeny_008.fasta      Progeny_014.fasta    Progeny_020.fasta
Progeny_003.fasta      Progeny_009.fasta      Progeny_015.fasta    P1.fasta
Progeny_004.fasta      Progeny_010.fasta      Progeny_016.fasta    P2.fasta
Progeny_005.fasta      Progeny_011.fasta      Progeny_017.fasta
Progeny_006.fasta      Progeny_012.fasta      Progeny_018.fasta
```

Next, you can finish the genotyping procedure by either (i) executing the integrated pipeline RADtyping .pl (default parameters):

```
mgb@M:~/djz/$ perl RADtyping.pl -p1 proc_data/P1.fasta -p2 proc_data/P2.fasta -l 27
```

Ten output files are created in two directories, “ref” and “genotype”.

```
mgb@M:~/djz/ref$ ls
ref_codom    ref_dom    HQ_ref_codom    HQ_ref_dom
mgb@M:~/djz/genotype$ ls
all_codom    all_dom    codom_JM    dom_JM    poly_dodom    poly_dom
```

Or, (ii) going through a few executions step by step:

Step1: Run ref_build.pl to reconstruct the representative reference sites.

```
mgb@M:~/djz/$ perl ref_build.pl -p1 proc_data/P1.fasta -p2 proc_data/P2.fasta
```

Reference sites are stored in two output files, “ref/ref_codom” and “ref/ref_dom”.

Step2: Run ref_filter.pl to obtain HQ reference sites.

```
mgb@M:~/djz/$ perl ref_filter.pl -m P
```

HQ sites are stored in two output files, “ref/HQ_ref_codom” and “ref/HQ_ref_dom”.

Step3: Run reads_map.pl to map HQ reads to HQ reference sites.

```
mgb@M:~/djz/$ perl reads_map.pl -l 27
```

All mapping results are stored in a directory named “reads_mapping”:

```
mgb@M:~/djz/reads_mapping$ ls
Progeny_001    Progeny_007    Progeny_013    Progeny_019
Progeny_002    Progeny_008    Progeny_014    Progeny_020
Progeny_003    Progeny_009    Progeny_015    P1
Progeny_004    Progeny_010    Progeny_016    P2
Progeny_005    Progeny_011    Progeny_017
Progeny_006    Progeny_012    Progeny_018
```

Step4: Run codom_calling.pl for performing codominant genotyping.

```
mgb@M:~/djz/$ perl codom_calling.pl -a 0.05 - p 0.8
```

Codominant genotypes are stored in two output files, "genotype/all_codom" and "genotype/poly_codom".

Step5: Run dom_calling.pl for performing dominant genotyping.

```
mgb@M:~/djz/$ perl dom_calling.pl - p 0.8
```

Dominant genotyes are stored in two output files, "genotype/all_dom" and "genotype/poly_dom".

Step6: Run joinmap_trans.pl to transform the genotyping results in a Joinmap-ready format.

```
mgb@M:~/djz/$ perl joinmap_trans.pl
```

Two joinmap-format files are "genotype/codom_JM" and "genotype/dom_JM".

Reference

[1] Catchen, J., Amores, A., Hohenlohe, P., Cresko, W. & Postlethwait, J. Stacks: building and genotyping loci *de novo* from short-read sequences. *G3: Genes, Genomes, Genetics* **1,** 171-182 (2011).

[2] Li, R. *et al.* SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* **25,** 1966-1967 (2009).

[3] Zerbino, D. R. *et al.* Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* **18,** 821-829 (2008)

[4] Dou, J. *et al.* Reference-free SNP calling: Improved accuracy by preventing incorrect calls from repetitive genomic regions. *Biol. Direct* **7,** 17 (2012).

[5] Hohenlohe, P. A. *et al.* Population genomics of parallel adaptation in threespine stickleback using sequenced RAD tags. *PLoS Genet*. **6**, e1000862 (2010).