Reference-based 2bRAD scripts: a manual

October 14, 2014

These are the printouts from scripts when run without arguments, in order of their appearance in the walkthrough.

```
ngs_concat.pl :

concatenates files by matching pattern in their names

arg1: common pattern for files
arg2: perl-like pattern in the filename to recognize,
        use brackets to specify the unique part, as in
      "FilenameTextImmediatelyBeforeSampleID(.+)FilenameTextImmediatelyAfterSampleID"

Example (to concatenate files names like Sample_Pop1_L1.fastq, Sample_Pop1_L2.fastq):
ngs_concat.pl 'Sample' 'Sample_(.+)_L'
```

```
2bRAD_trim_launch_dedup.pl :

Prints out list of calls to trim2bRAD_2barcodes_dedup.pl, one for
each fastq file, to to trim and deduplicate raw 2bRAD reads.

See help for trim2bRAD_2barcodes_dedup.pl for more details.

prints to STDOUT

Arguments:
arg1, required: glob to fastq files
site=[pattern] perl-style pattern of the restriction site to recognise, in single quotes.
            default is BcgI: \'.{12}CGA.{6}TGC.{12}|.{12}GCA.{6}TCG.{12}\'
            if you need AlfI, here is how to define it: \'.{12}GCA.{6}TGC.{12}\'
adaptor=[dna sequence] adaptor sequence to look for on the far end of the read.
            Default AGATC
sampleID=[integer] the position of name-deriving string in the file name
                  if separated by underscores, such as:
                  for input file Sample_RNA_2DVH_L002_R1.cat.fastq
                  specifying arg2 as \'3\' would create output
                  file with a name \'2DVH.trim'
barcode2=[integer] length of the in-line barcode immediately following
                     the restriction fragment. Default 0.

Example:
2bRAD_trim_launch_dedup.pl fastq sampleID=3 > trims
```

---

trim2bRAD_2barcodes_dedup.pl :

```
This script does three things:

- Filters 2bRAD fastq reads to leave only those with a 100% matching restriction site,
degenerate 5'-leader, secondary 3'-barcode, and adaptor on the far 3'end,
trims away everything except the IIb-fragment itself;

- Deduplicates: removes all but one read sharing the same 64-fold degenerate
leader, the first 34 bases of the insert sequence, and secondary barcode
(this results in 128-fold dynamic range: 64-fold degeneracy x 2 strand orientations);

- Splits reads into separate files according to secondary barcode.

Writes trimmed fastq files named according to the secondary barcodes detected.

Arguments:
input=[fastq filename]
site=[perl-style pattern for restrictase site]
           Default is BcgI: \'.{12}CGA.{6}TGC.{12}|.{12}GCA.{6}TCG.{12}\'
bc2=[perls-style barcode pattern] in-line barcode that immediately follows the RAD
fragment,
           default \'[ATGC]{4}\'
adaptor=[far-end adaptor sequence] default AGATC
clip=[integer] number of bases to clip off the ends of the reads, default 0
minBCcount=[integer] minimum count per in-line barcode to output a separate file.
                            Default 100000.
sampleID=[integer] the position of name-deriving string in the file name
                             if separated by underscores, such as:
                             for input file Sample_RNA_2DVH_L002_R1.cat.fastq
                             specifying arg2 as \'3\' would create output
                             file with a name \'2DVH.trim'

Example:
trim2bRAD_2barcodes_dedup.pl input=Myproject_L8_G3.fastq sampleID=2
```

---

concatFasta.pl :

```
concatenates fasta records in a draft genome reference into \"pseudo-chromosomes\"
(either a specified number of them or as many as needed to
split the genome into chunks just over 100G bases in length)

Do this to your genotyping reference to improve the speed and memory usage
of the UnifiedGenotyper (GATK)

Arguments:

fasta=[file name] : fasta file name
num=[integer]     : number of pseudo-chromosomes, default 10

Output:

[fasta filename base]_cc.fasta : new fasta file
[fasta filename base]_cc.tab   : table of original sequence IDs, their coordinates
                                 in pseudo-chromosomes, and full original header lines
Example: concatFasta.pl fasta=mygenome.fasta
```

---

retabvcf.pl :

```
rewrites coordinates of variants in a VCF file according to a table
of locations of original contigs in pseudo-chromosomes
(created by concatFasta.pl)

Arguments:

vcf=[file name] : vcf file to re-coordinate
tab=[file name] : tab-delimited table of contig locations in the form
                  contig-chromosome-start-end

Output:
VCF with reformatted coordinates (printed to STDOUT)

Example:
retabvcf.pl vcf=gatk_after_vqsr.vcf tab=where/genome/is/mygenome_cc.tab > retab.vcf
```

replicatesMatch.pl : (version 0.2, September 17, 2014)

```
Selects variants that have identical genotypes among replicates.

A genotype should be identical or missing; the fraction of allowed
missing genotypes is controlled by missing=  argument.

Arguments:

vcf=[file name]  input vcf file
replicates=[file name] - a two column tab-delimited table listing
                 pairs of samples that are replicates (at least 3 pairs)

matching=[float] required fraction of matching genotypes
                 (missing counts as match if there are other non-missing matching pairs),
                 default 1 (all must match)

missing=[float]  allowed fraction of missing genotypes, default 0.25

altonly=[1|0]    only output SNPs showing alternative alleles (not necessarily
                 polymorphic among genotyped samples). Default 0

polyonly=[1|0]   output only those passing SNPs that are polymorphic among
                 replicated samples; for variant quality score recalibration
                 (VQSR) in GATK. Overrides altonly setting. Default 0

min.falt=[float] minimum minor allele frequency. Default 1/(number of replicate pairs)

max.falt=[float] maximum minor allele frequency. Default 0.35

Example:
replicatesMatch.pl vcf=round2.names.vcf \
replicates=clonepairs.tab polyonly=1 max.falt=0.4 > vqsr.vcf
```

recalibrateSNPs_gatk.pl :

Non-parametric variant quality recalibration based on INFO fields
in the set of variants that are reproducibly genotyped among replicates
(output of replicatesMatch.pl)

This script is for reference-based pipeline. Use recalibrateSNPs.pl for
de novo pipeline.

Three parameters (INFO field) are assessed for each SNP:
- fisher-strand (FS)
- mapping quality (MQ)
- quality by depth (QD)
- depth (DP)

The script computes the product of quantiles for different INFO fields, determines
the quantiles of the result within the 'true' set (JOINT quantiles),
and then computes the new quality scores in the main vcf file.

These scores are supposed to correspond to the probability (x100) that the
SNPs comes from the same distribution as the 'true' SNPs.

Output:

- Recalibrated VCF (printed to STDOUT) with QUAL field replaced by the new score
  minus 0.1 (for easy filtering)

- a table (printed to STDERR) showing the \"gain\" at each recalibrated quality
  score setting, which is excess variants removed when filtering at this quality score.
  For example, if 40% of all variants are removed at the quality score 15, the gain
  is 40 - 15 = 35% (i.e., in addition to removing 15% of the 'true' variants,
  additional 35% of the dataset, likely corresponding to wrong variants, is removed).
  The optimal filtering score is the one giving maximum gain. Try different combinations
  of the four possible filters to find the one maximizing the gain.

Arguments:

```
vcf=[file name]  : vcf file to be recalibrated

true=[file name] : vcf file that is subset of the above, with SNPs that are considered
                   true because they show matching and polymorphic genotypes in
replicates
                   (replicatesMatch.pl polyonly=1).

           -nodp : do not use DP
           -nofs : do not use FS
           -nomq : do not use MQ
           -noqd : do not use QD
```

Example:

```
recalibrateSNPs_gatk.pl vcf=round2.names.vcf true=vqsr.vcf > recal.vcf
```

---

filterStats_gatk.pl :

Calculates quantiles for three filtering criteria using VCF file:
- fisher-strand (FS)
- mapping quality (MQ)
- depth (DP)

```
For MQ, low values are bad and high values are good;
for FS, high values are bad;
for DP, both high and low values are bad

Arguments:

vcf=[file name]  vcf file to analyze

Output:
Tables of quantiles (printed to STDOUT).

Example:
filterStats_gatk.pl vcf=round2.names.vcf

NOTE: only use this to select your filtering criteria if you don't have
genotyping replicates.
```

---

repMatchStats.pl :

```
Summarizes genotypic match between replicates in a vcf file.

Arguments:

       vcf=[file name] : input vcf file
replicates=[file name] : a two column tab-delimited table listing
                         pairs of samples that are replicates

Output:
A table (printed to STDOUT) of the following form:

pair      gtyped  match      [ 00  01  11 ] HetMatch HomoHetMism HetNoCall HetsDiscRate
K210:K212  7328  7169(97.8%) [ 78% 17% 5% ]  1200         135        5         0.94
K212:K213  7369  7117(96.6%) [ 78% 17% 5% ]  1202         179        4         0.93

The first four columns are self-evident;
the last four columns show how good is the match between heterozygote calls
(HetNoCall being a heterozygote in one and missing data in another replicate).
The last column is the most important: it is the heterozygote discovery rate,
the fraction of all heterozygotes discovered in each replicate.

Example: repMatchStats.pl vcf=filt.recode.vcf replicates=clonepairs.tab
```

---

thinner.pl :

```
Leaves only one SNP per given interval, the one with the highest minor allele frequency.

Arguments:

vcf=[file name]    : vcf file name
interval=[integer] : interval length, default 40 (for 2bRAD tags)

Output:  thinned VCF, printed to STDOUT

Example: thinner.pl vcf=recal.vcf > thin.vcf

NOTE: do not thin your variants if you want to calculate Tajima's D!
```

```
vcf2genepop.pl :

converts VCF to multiallelic GENEPOP, preserves chromosome and position info

Arguments:
vcf=[filename] : vcf file to convert
   pops=[list] : comma-separated perl patterns to determine population
                 affiliation of samples in the VCF file based on sample names

Output:
GENEPOP formatted genotypes, printed to STDOUT

Example:
vcf2genepop.pl vcf=filtered.vcf pops=O,K,M,S > filtered.gen
```
