Pipeline for *de novo* Targeted Capture

October 6, 2014

Contributors: Sonal Singhal, Ke Bi, Tyler Linderoth

For questions or to report bugs, please contact Ke Bi (kebi@berkeley.edu)

Reference:
[1]. Singhal S. 2013. De novo transcriptomic analyses for non-model organisms: an evaluation of methods across a multi-species data set. Molecular Ecology Resources 13:403-416.
[2]. Bi K, Linderoth T, Vanderpool D, Good JM, Nielsen R and Moritz C. 2013. Unlocking the vault: next-generation museum population genomics. Molecular Ecology 22:6018-6032.
[3]. Bi K, Vanderpool D, Singhal S, Linderoth T, Moritz C and Good JM. 2012. Transcriptome-based exon capture enables highly cost-effective comparative genomic data collection at moderate evolutionary scales. BMC Genomics 13: e403.

The pipelines are deposited in https://github.com/MVZSEQ/denovoTargetCapture

---

Scripts included in this pipeline:

Use “chmod +x script” to make each of these perl scripts executable.

- Exon/exome/sequence capture dataset
  Use 1->2->3->4->5->6->7->8 when no reference genome is available;
  Use 1->2->7->8 when a reference genome is available

- Genomic dataset (with a pre-existing genome or *de novo* genome scaffolds)
  Use 1->2->7

- *de novo* transcriptome dataset
  Use 1->2->3->4->5->7

- Single RAD/GBS dataset
  Use 1->2->3->RAD/GBS Tag filtering->7

- ddRAD/ddGBS dataset
  Use 1->2->ddRAD pipelines->7

- UCE dataset
  Use 1->2->UCE pipelines->7

1-PreCleanup: Reformats raw sequencing reads from Illumina HiSeq or MiSeq for *2-ScrubReads*. Specifically, in this step we will remove reads that did not pass the Illumina quality control filters and modify the sequence identifiers.

Dependencies:
FastQC: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Input:
Raw sequence data files are grouped and saved in folders named by their sample IDs. For instance, three libraries (CGRL_index1, CGRL_index15, CGRL_index40) are saved under "/home/ke/Desktop/SeqCap/data/rawdata/library/". Compressed fastq sequence files are saved in each of these folders.

Fastq files use the following naming scheme:
<sample name>_<barcode sequence>_L<lane (0-padded to 3 digits)>_R<read number>_<set number (0-padded to 3 digits)>.fastq.gz

For example, in "CGRL_index15_CGACCTG_L006_R1_001.fastq.gz":

sample name: CGRL_index15
barcode sequence: CGACCTG
lane (0-padded to 3 digits): 006
read number: 1
set number (0-padded to 3 digits): 001

#Make a new folder called "raw" under "~/Desktop/SeqCap/data/rawdata/":
ke@NGS:~/Desktop/SeqCap/data/rawdata$ mkdir raw

#Copy all these compressed fastq files from each folder (CGRL_index1, CGRL_index15, CGRL_index40) to “raw”:

```
ke@NGS:~/Desktop/SeqCap/data/rawdata$ cp library/CGRL_index*/*.gz raw/
```

#Check data files in “raw”:

```
ke@NGS:~/Desktop/SeqCap/data/rawdata$ ls raw/*
CGRL_index15_CGACCTG_L006_R1_001.fastq.gz
CGRL_index15_CGACCTG_L006_R2_001.fastq.gz
CGRL_index1_TCGCAGG_L006_R1_001.fastq.gz
CGRL_index1_TCGCAGG_L006_R2_001.fastq.gz
CGRL_index40_TTCGCAA_L006_R1_001.fastq.gz
CGRL_index40_TTCGCAA_L006_R2_001.fastq.gz
```

Commands:

#cd to the working directory:

```
ke@NGS:~/Desktop/SeqCap/data/rawdata$ cd ..
```

#run 1-PreCleanup with fastq evaluation

```
ke@NGS:~/Desktop/SeqCap/data$ 1-PreCleanup
~/Desktop/SeqCap/data/rawdata/raw/ fastqc
```

Output:

Three new folders will be created under “~/Desktop/SeqCap/data/rawdata/raw/”:
“pre-clean”
“combined”
“pre-clean/evaluation”

- Folder “pre-clean” contains reformatted raw fastq reads.

```
CGRL_index1_R1.fq
CGRL_index1_R2.fq
CGRL_index15_R1.fq
CGRL_index15_R2.fq
CGRL_index40_R1.fq
CGRL_index40_R2.fq
```

- Folder “combined” contains merged, compressed, fastq data files (not used by the following pipeline).

```
CGRL_index1_TCGCAGG_L006_R1.fastq.gz
CGRL_index1_TCGCAGG_L006_R2.fastq.gz
CGRL_index15_CGACCTG_L006_R1.fastq.gz
CGRL_index15_CGACCTG_L006_R2.fastq.gz
CGRL_index40_TTCGCAA_L006_R1.fastq.gz
CGRL_index40_TTCGCAA_L006_R2.fastq.gz
```

- Folder “evaluation” contains fastQC results for each data file.

```
CGRL_index1_R1.fq_fastqc/
```

```
CGRL_index1_R2.fq_fastqc/
CGRL_index15_R1.fq_fastqc/
CGRL_index15_R2.fq_fastqc/
CGRL_index40_R1.fq_fastqc/
CGRL_index40_R2.fq_fastqc/
```

Questions:
1. Check the sequence identifiers and the number of reads in fastq files before and after running *1-PreCleanup* and compare the results.
2. Check the fastQC evaluation results for the raw data

2-ScrubReads: Clean up raw data, which includes trimming for quality, removing adapters, merging overlapping reads, removing duplicates and reads sourced from contamination

Dependencies:
cutadapt: http://code.google.com/p/cutadapt/
COPE: http://sourceforge.net/projects/coperead/
Bowtie2: http://sourceforge.net/projects/bowtie-bio/files/bowtie2/
FastQC: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
FLASh-modified: modified version of FLASh by Filipe G. Vieira. https://github.com/MVZSEQ/Exon-capture
Trimmomatic: http://www.usadellab.org/cms/?page=trimmomatic

Input:
1. Reformatted fastq files created by *1-PreCleanup*:
#Check the raw data files:

```
ke@NGS:~/Desktop/SeqCap/data/rawdata/raw/pre-clean$ ls *.fq
CGRL_index1_R1.fq
CGRL_index1_R2.fq
CGRL_index15_R1.fq
CGRL_index15_R2.fq
CGRL_index40_R1.fq
CGRL_index40_R2.fq
```

2. A fasta file that contains adapter sequences:
#Check the format of adapter sequence file:

```
ke@NGS:~/Desktop/SeqCap/denovoTargetCapture/associated_files $ less -S
Adapters.fasta
>P7_index1
CAAGCAGAAGACGGCATACGAGATcctgcgaGTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT
>P7_index2
CAAGCAGAAGACGGCATACGAGATtgcagagGTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT
......
>P5_index1
```

```
AATGATACGGCGACCACCGAGATCTACACcctgcgaACACTCTTTCCCTACACGACGCTCTTCCGATCT
>P5_index2
AATGATACGGCGACCACCGAGATCTACACtgcagagACACTCTTTCCCTACACGACGCTCTTCCGATCT
......
```

Note: The header of each adapter sequence has to be named strictly as "**P7_index**N" or "**P5_index**N". N is the number of index. It is OK to put all adapters in this file but your libraries only use a subset of them.

3. Library info file (Tab-delimited txt file):
#Check the format of Library info file:

```
ke@NGS:~/Desktop/SeqCap/denovoTargetCapture/associated_files $ less -S libInfo.txt
```

```
library         P7      P5
CGRL_index1     1
CGRL_index15    15
CGRL_index40    40
```

leave the "P5" column blank if you only have indexes in P7 adapters in the libraries.

4. Contaminant file:
Escherichia coli (+ human + other genome resources if desired) genome in fasta format.
This file (e_coli_K12.fasta) is saved in "~/Desktop/SeqCap/denovoTargetCapture/associated_files/ecoli/"

Commands:
#Make a new folder called "cleaned_data" in "~/Desktop/SeqCap/data/":

```
ke@NGS:~/Desktop/SeqCap/data$ mkdir cleaned_data
```

#Run 2-ScrubReads:

```
ke@NGS:~/Desktop/SeqCap/data$ 2-ScrubReads -f
~/Desktop/SeqCap/data/rawdata/raw/pre-clean/ -o
~/Desktop/SeqCap/data/cleaned_data/ -a
~/Desktop/SeqCap/denovoTargetCapture/associated_files/Adapters.fasta -b
~/Desktop/SeqCap/denovoTargetCapture/associated_files/libInfo.txt -t
/home/ke/Desktop/SeqCap/programs/Trimmomatic-0.32/trimmomatic-0.32.jar -c
~/Desktop/SeqCap/denovoTargetCapture/associated_files/ecoli/e_coli_K12.fasta -e
200 -m 15 -z
```

Note: I use default values for most of the arguments. Users should adjust these parameters when processing the real datasets.

Output:
1. In "~/Desktop/SeqCap/data/cleaned_data/", six .txt files per library are produced:

For example for library CGRL_index1, the six files are:
CGRL_index1_1_final.txt (left reads)
CGRL_index1_2_final.txt (right reads)
CGRL_index1_u_final.txt (merged or unpaired reads)
CGRL_index1.contam.out (headers of reads aligned to bacteria)
CGRL_index1.duplicates.out (headers of duplicated reads)
CGRL_index1.lowComplexity.out (headers of low complexity reads)

2. In "~/Desktop/SeqCap/data/cleaned_data/evaluation/", you can find fastQC results for cleaned reads from each library.

Questions:
1. Check the %reads that are exact duplicates, %reads that are likely derived from microbial genome and %reads that contain low complexity.
2. Check the fastQC evaluation results of cleaned reads and then compare them to those of raw reads. Is the quality improved?

**3-GenerateAssemblies*: Assemble sequence capture data using ABySS.

We use a multiple-kmer approach to assemble our data. If there is even coverage and even polymorphism levels across the assembled genome, there should (in theory) be one k-mer that best assembles the data. In reality, coverage and polymorphism vary across captured loci, and using multiple k-mers is a way to bet hedge and get good assemblies for all loci. In assembling your data, it is important to consider which samples to use in your assembly. Ideally, you could assemble across multiple individuals to increase your read depth, and thus, assembly contiguity and continuity. However, for many projects, more individuals can also mean increased polymorphism. While we have found the assemblers are more robust to polymorphism than the program writers themselves often suggest, increased polymorphism does lead to shorter contigs and increased misassemblies. With these sample data, we assembled across all in-group samples – this seemed like the best balance between having enough data to power assembly while not introducing too much polymorphism.

Dependencies:
ABySS (compiled with OpenMPI and Google sparsehash):
http://www.bcgsc.ca/platform/bioinfo/software/abyss

Input:
Concatenated cleaned reads from libraries that you would like to assemble together. The libraries to be assembled together have to be genetically similar: ideally, samples from the same population. In this example we want to assemble CGRL_index1, CGRL_index15 and CGRL_index40 together.

#Make a new folder called "raw_assembly" under "~/Desktop/SeqCap/data/":

```
ke@NGS:~/Desktop/SeqCap/data$ mkdir raw_assembly
```

#Concatenate cleaned reads and save them in "raw_assembly":

```
ke@NGS:~/Desktop/SeqCap/data$ cat cleaned_data/CGRL_index*_1_final.txt >
raw_assembly/combined_1_final.txt
ke@NGS:~/Desktop/SeqCap/data$ cat cleaned_data/CGRL_index*_2_final.txt >
raw_assembly/combined_2_final.txt
ke@NGS:~/Desktop/SeqCap/data$ cat cleaned_data/CGRL_index*_u_final.txt >
raw_assembly/combined_u_final.txt
```

#Inside "raw_assemblies" make a new folder "results":

```
ke@NGS:~/Desktop/SeqCap/data$ mkdir raw_assembly/results
```

Commands:

#Run ABySS on two processors using kmer sizes of 21, 31, 41, 51, 61, and 71.

```
 ke@NGS:~/Desktop/SeqCap/data$ 3-GenerateAssemblies abyss -reads
~/Desktop/SeqCap/data/raw_assembly/ -mpi /usr/bin/mpirun -out
~/Desktop/SeqCap/data/raw_assembly/results/ -kmer 21 31 41 51 61 71 -np 2
```

Note: Your labtop will not be able to handle memory intensive ABySS assemblies.

Output:

There are a lot of intermediate files created in "~/Desktop/SeqCap/data/raw_assembly/results/combined/".

#To show the assemblies that we need for the next step:

```
ke@NGS:~/Desktop/SeqCap/data$ cd raw_assembly/results/combined/
ke@NGS:~/Desktop/SeqCap/data/raw_assembly/results/combined$ ls *-contigs.fa
combined_k21_cov_default-contigs.fa
combined_k31_cov_default-contigs.fa
combined_k41_cov_default-contigs.fa
combined_k51_cov_default-contigs.fa
combined_k61_cov_default-contigs.fa
combined_k71_cov_default-contigs.fa
```

#Combine all the raw assemblies and write the result to a new file called "all_assemblies.fasta":

```
ke@NGS:~/Desktop/SeqCap/data/raw_assembly/results/combined$ cat
combined_*_cov_default-contigs.fa > all_assemblies.fasta
```

#Make a new folder called "merge_assemblies" under "~/Desktop/SeqCap/data/":

```
ke@NGS:~/Desktop/SeqCap/data $ mkdir merge_assemblies
```

#Copy "all_assemblies.fasta" into "merge_assemblies/":

ke@NGS:~/Desktop/SeqCap/data $ cp raw_assembly/results/combined/all_assemblies.fasta merge_assemblies

---

4-FinalAssembly: Combining assembled contigs across multiple k-mers to generate a final assembly introduces a lot of redundancy into the final assembly. To address this, we use a lightweight assembler cap3 and other programs (blat, cd-hit-est) to merge contigs and to remove redundancies.

Dependencies:
CAP3: http://seq.cs.iastate.edu/cap3.html
blat: http://users.soe.ucsc.edu/~kent/src/
cd-hit-est: https://code.google.com/p/cdhit/downloads/list

Input:
Concatenated raw assemblies "~/Desktop/SeqCap/data/merge_assemblies/all_assemblies.fasta" produced by *3-GenerateAssemblies*

Commands:
ke@NGS:~/Desktop/SeqCap/data$ 4-FinalAssembly -a ~/Desktop/SeqCap/data/merge_assemblies/ -c 1000

Note: when analyzing real data, users should test these parameters (-d -e -b) for optimal results.

Output:
Several files are created in "~/Desktop/SeqCap/data/merge_assemblies/". The data file that we need for the next step is "all_assemblies.fasta.final".

# Rename "all_assemblies.fasta.final":
ke@NGS:~/Desktop/SeqCap/data/merge_assemblies$ mv all_assemblies.fasta.final all_assemblies_final.fasta

---

5-FindTargets: identify contigs that are stemmed from the targeted loci and use these contigs as a reference (aka. a pseudo-reference)

Here, we suggest taking a very conservative approach to define the reference genome against which you will align your reads. You will likely get many multiples more contigs than loci you targeted. Some of these might be junk; some might be real. Rather than try to identify which of the extraneous contigs are junk or real, we suggest using only those contigs which match to the original targets. To do so, we implement a BLAST approach, which identifies which contig has the best-hit match to one's targeted loci.

Dependencies:
blastn (BLAST+): ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
cd-hit-est: https://code.google.com/p/cdhit/downloads/list

Input:
1. "~/Desktop/SeqCap/data/merge_assemblies/all_assemblies_final.fasta" produced by *4-FinalAssembly*.

2. Targeted loci fasta file:
"~/Desktop/SeqCap/denovoTargetCapture/original_target /targeted_loci.fasta" contains loci/genes/exons from which probes are designed.

Commands:
#Make a new folder called "in_target_assemblies": under "~/Desktop/SeqCap/data/":
ke@NGS:~/Desktop/SeqCap/data$ mkdir in_target_assemblies

#Run *5-FindingTargets* in "~/Desktop/SeqCap/data/in_target_assemblies/":
ke@NGS:~/Desktop/SeqCap/data/in_target_assemblies$ 5-FindingTargets -t ~/Desktop/SeqCap/denovoTargetCapture/original_target/targeted_loci.fasta -a ~/Desktop/SeqCap/data/merge_assemblies/all_assemblies_final.fasta -o in_target.fasta -e in_target.captured

Output:
Two files are created and stored in "~/Desktop/SeqCap/data/in_target_assemblies/":

1. "in_target.fasta": A fasta sequence file containing contigs that are stemmed from the targeted loci.

2. "in_target.captured": A txt file containing percent captured for each target (tab-delimited).

6-AssemblyEvaluation (Optional): function "BASIC" evaluates the quality of in-target assemblies by reporting basic stats: mean, median, total length, gc%, N50 etc. It also generates a distribution of contigs by binned lengths.

In reality, "BASIC" can evaluates quality of any assemblies.

This script also assesses the quality of transcriptome/targeted capture assemblies form several other aspects. For example:

"COVERAGE" calculates error rate, average quality score of the aligned bases and its variance/std, and average base coverage. Users need to generate alignment first.
"FIX" fixes assembly errors. Users need to generate alignment first.

For more details please execute "*6-AssemblyEvaluation*" in a terminal window.

6-AssemblyEvaluation BASIC
Input:
"~/Desktop/SeqCap/data/in_target_assemblies/in_target.fasta" produced by *5-FindTargets*.

Commands:
```
ke@NGS:~/Desktop/SeqCap/data $ 6-AssemblyEvaluation BASIC -a
~/Desktop/SeqCap/data/in_target_assemblies
```

Output:
Two files are created in "~/Desktop/SeqCap/data/in_target_assemblies/":

1. "in_target.hist": distribution of contigs by binned lengths
#Display first few lines of the file:
```
ke@NGS:~/Desktop/SeqCap/data/in_target_assemblies$ head in_target.hist
200:299      1026
300:399      242
400:499      73
500:599      17
600:699      7
700:799      0
800:899      0
900:999      1
1000:1099    0
1100:1199    1
```

2. "basic_evaluation.out": results of assembly evaluation
#Display first few lines of the file:
```
ke@NGS:~/Desktop/SeqCap/data/in_target_assemblies$ head basic_evaluation.out
```

Note: users might want to compare the metric between in_target assemblies to original targeted loci and see how much flanking are captured and assembled. Under most circumstances, the mean, median, N50 etc should be much higher in in_target assemblies than in original targeted loci.
The example dataset that is used for the purpose of demonstration, however, should not show this pattern indicated above since it is assembled from a tiny fraction of data.

7-Alignment: aligning cleaned reads against the pseudo-reference using Novoalign

Novoalign "is an aligner for single-ended and paired-end reads from the Illumina. Novoalign finds global optimum alignments using full Needleman-Wunsch algorithm with affine gap penalties."

"Question: How does Novoalign compare to programs like BWA, Bowtie, ELAND and BFAST?
Answer:
Novoalign was designed to be an accurate short read aligner that combines fast K-mer index searching with dynamic programming. In terms of speed Novoalign will be slower than Burrows-Wheeler transform aligners e.g. BWA, Bowtie and in some cases faster than BFAST. In terms of accuracy Novoalign is in most cases more sensitive than these tools because it uses full dynamic programming to find the best alignment of a short read to a genome sequence."

According to Heng Li, author of SAMTools & MAQ, Novoalign "is the most accurate aligner to date".

Dependencies:
Novoalign: http://www.novocraft.com/main/downloadpage.php
SAMTools: http://sourceforge.net/projects/samtools/files/samtools/

Input:
1. A pseudo-reference genome, "~/Desktop/SeqCap/data/in_target_assemblies/in_target.fasta", generated by *5-FindTargets*:
#make a new directory called "reference" under "~/Desktop/SeqCap/data/":
```
ke@NGS:~/Desktop/SeqCap/data$ mkdir reference
```

#Copy "in_target.fasta" to "~/Desktop/SeqCap/data/reference/":
```
ke@NGS:~/Desktop/SeqCap/data$ cp in_target_assemblies/in_target.fasta reference/
```

2. Cleaned reads generated by *2-ScrubReads:*
Cleaned reads are saved in "~/Desktop/SeqCap/data/cleaned_data/".
#Take a look at these reads:
```
ke@NGS:~/Desktop/SeqCap/data/cleaned_data$ ls *.txt
CGRL_index1_1_final.txt
CGRL_index1_2_final.txt
CGRL_index1_u_final.txt
CGRL_index15_1_final.txt
CGRL_index15_2_final.txt
CGRL_index15_u_final.txt
CGRL_index40_1_final.txt
CGRL_index40_2_final.txt
CGRL_index40_u_final.txt
```

Commands:

```
#Make a new folder called "alignment" under "~/Desktop/SeqCap/data/":
ke@NGS:~/Desktop/SeqCap/data$ mkdir alignment

#Run 7-Alignment:
ke@NGS:~/Desktop/SeqCap/data$ 7-Alignment -f
~/Desktop/SeqCap/data/reference/in_target.fasta -r
~/Desktop/SeqCap/data/cleaned_data/ -o ~/Desktop/SeqCap/data/alignment/ -i
200 -v 20 -t 90
```

Note: do not set t for alignment of very divergent genomes.

Output:
BAMS and indexed bam files.
#Take a look at these files:

```
ke@NGS:~/Desktop/SeqCap/data/alignment$ ls
CGRL_index1_sorted.bam
CGRL_index1_sorted.bam.bai
CGRL_index15_sorted.bam
CGRL_index15_sorted.bam.bai
CGRL_index40_sorted.bam
CGRL_index40_sorted.bam.bai
```

8-ExonCaptureEvaluation (Optional): Function "Evaluation" provides evaluation for capture efficiency: %reads mapped, %target captured, average sequence depth, etc.

Note: %reads mapped (specificity), %target captured (sensitivity), and average sequence depth are typically reported in papers.

Dependencies:
SAMTools: http://sourceforge.net/projects/samtools/files/samtools/
BEDTools: http://bedtools.readthedocs.org/en/latest/content/installation.html

8-ExonCaptureEvaluation Evaluation
Input:
1. A pseudo-reference "in_target.fasta" generated by *5-FindTargets*:
You can find this file in "~/Desktop/SeqCap/data/reference/".

2. Cleaned reads generated by *2-ScrubReads*:
These reads are located in "~/Desktop/SeqCap/data/cleaned_data/":
CGRL_index1_1_final.txt
CGRL_index1_2_final.txt

CGRL_index1_u_final.txt
CGRL_index15_1_final.txt
CGRL_index15_2_final.txt
CGRL_index15_u_final.txt
CGRL_index40_1_final.txt
CGRL_index40_2_final.txt
CGRL_index40_u_final.txt

3. Raw reads generated by *1-PreCleanup*:
These data are located in "~/Desktop/SeqCap/data/rawdata/raw/pre-clean/":
CGRL_index1_R1.fq
CGRL_index1_R2.fq
CGRL_index15_R1.fq
CGRL_index15_R2.fq
CGRL_index40_R1.fq
CGRL_index40_R2.fq

4. All bam (alignment) files generated by *7-Alignment*:
The bams (sorted and indexed) are located in "~/Desktop/SeqCap/data/alignment/":
CGRL_index1_sorted.bam
CGRL_index1_sorted.bam.bai
CGRL_index15_sorted.bam
CGRL_index15_sorted.bam.bai
CGRL_index40_sorted.bam
CGRL_index40_sorted.bam.bai

5. A .bed file generated by *9-preFiltering* (optional)
A BED file (.bed) is a tab-delimited text file that defines a feature track of each locus. In this case, this file defines targeted region in each assembled contig.

For example if the length of contig125 is 1000bp, but the targeted region starts from position 120 and ends by 350, then the correct expression is:

Contig125 119 350 (note: in bed the start position is one less than it's actual value)

For more details of BED format please go to:
http://www.broadinstitute.org/igv/BED

Commands:
#Make a new folder called "ExonCapEval" under "~/Desktop/SeqCap/data/":
ke@NGS:~/Desktop/SeqCap/data$ mkdir ExonCapEval

#Run *8-ExonCaptureEvaluation*:
ke@NGS:~/Desktop/SeqCap/data$ 8-ExonCaptureEvaluation Evaluation -genome ~/Desktop/SeqCap/data/reference/in_target.fasta -cleanDir

```
~/Desktop/SeqCap/data/cleaned_data/ -rawDir
~/Desktop/SeqCap/data/rawdata/raw/pre-clean/ -bamDir
~/Desktop/SeqCap/data/alignment/ -InstrID HS -resDir
~/Desktop/SeqCap/data/ExonCapEval/ -readlen 100
```

Note: If you just evaluate how targeted regions worked, you should provide a bed file (generated by 9-preFiltering) while running 8-ExonCaptureEvaluation Evaluation.

Output:
"data_metrics.txt" under "~/Desktop/SeqCap/data/ExonCapEval/"
You can use "less" to check the results reported in this file.

9-preFiltering: "*9-preFiltering* bed" generates a bed for exonic region(s) from each contig in in-target assemblies (aka. the reference) and a bed for all assembled contigs (start position is 0 and the end position is the length of the contig); "*9-preFiltering percentile*" produces a list of contigs that fall outside the desired coverage percentiles; "*9-preFiltering percentile*" also produces base coverage values at different level of percentile.

Dependencies:
Tie-Array-Packed-0.13: http://search.cpan.org/~salva/Tie-Array-Packed-0.13/lib/Tie/Array/Packed.pm

9-preFiltering bed:
Input:
1.Targeted loci:
"~/Desktop/SeqCap/denovoTargetCapture/original_target/targeted_loci.fasta";

2. "~/Desktop/SeqCap/data/reference/in_target.fasta" generated by *5-FindTargets*.

Commands:
#Make a new folder called "bed_files" under "~/Desktop/SeqCap/data/":
```
ke@NGS:~/Desktop/SeqCap/data$ mkdir bed_files
```

#cd to this folder:
```
ke@NGS:~/Desktop/SeqCap/data$ cd bed_files
```

#Run *9-preFiltering* bed:
```
ke@NGS:~/Desktop/SeqCap/data/bed_files$ 9-preFiltering bed
~/Desktop/SeqCap/denovoTargetCapture/original_target/targeted_loci.fasta
~/Desktop/SeqCap/data/reference/in_target.fasta
```

Output:
Two file under "~/Desktop/SeqCap/data/bed_files/":
1. "final.bed" is used as input for 9-preFiltering *percentile* and 8-ExonCaptureEvaluation *Evaluation*
2. "All_contig.bed" is used as input for 10-SNPcleaner.

9-preFiltering *percentile:*
Input:
1. Make a new folder called "pre-filtering" under "~/Desktop/SeqCap/data/" and cd to this folder:
```
ke@NGS:~/Desktop/SeqCap/data$ mkdir pre-filtering
ke@NGS:~/Desktop/SeqCap/data$ cd pre-filtering
```

2. In "~/Desktop/SeqCap/data/pre-filtering/", generate a merged, sorted bam for all samples:
```
ke@NGS:~/Desktop/SeqCap/data/pre-filtering$ samtools merge merge.bam
~/Desktop/SeqCap/data/alignment/*.bam
ke@NGS:~/Desktop/SeqCap/data/pre-filtering$ samtools sort merge.bam
merge_sorted
```

"~/Desktop/SeqCap/data/pre-filtering/merge_sorted.bam" is the input bam.

3. A bed file:
"~/Desktop/SeqCap/data/bed_files/final.bed" is generated by 9-preFiltering *bed*

Commands:
# Run 9-preFiltering *percentile* under "~/Desktop/SeqCap/data/pre-filtering/":
```
ke@NGS:~/Desktop/SeqCap/data/pre-filtering$ 9-preFiltering percentile -b
~/Desktop/SeqCap/data/pre-filtering/merge_sorted.bam -o CGRL -B
~/Desktop/SeqCap/data/bed_files/final.bed
```

Output:
In the folder "~/Desktop/SeqCap/data/pre-filtering/" there are a couple of files created:
1. "CGRL_gene_outside_percentile.txt" shows a list of contigs having coverage <X% or >Y% percentiles of the data. X and Y are defined by users. This file will be used in 10-SNPcleaner.
2. "CGRL_site_depth_percentile.txt" shows base coverage at different level of percentiles. The information in this file will be used by 10-SNPcleaner.
3. "CGRL_gene_depth_percentile.txt" shows average base coverage at different level of percentiles.
4. "CGRL_gene_depth.txt" shows average coverage of each contig. If you want to know more about empirical coverage distribution of your data then you take the coverage value from this file and use R to plot it.

5. "CGRL_site_depth.txt" shows per-base coverage of the data. You can plot it to get a sense of empirical distribution of base coverage.
6. "CGRL_gene_outside_sd_filter.txt": shows a list of contigs all outside N standard deviation of the mean. Users set N when running the command.

Note: users might want to perform filtering based on other criteria such as 3 standard deviations of the mean. However, this method usually requires a normal distribution of the data. In reality per-base depth of exon capture data rarely follows a normal distribution.

10-SNPcleaner: Raw variant filtering and generates a "keep" file for the following SNP/genotype calling by ANGSD. This script is mainly for filtering data at contig and site levels. Users need to perform individual-level filtering before running this script. See below for more details.

Before we call SNPs /genotypes and estimate allele frequencies using ANGSD, we usually employ three levels of filtering on the data sets in a hierarchical order: individual level, contig level and site level. The filters in each step of the hierarchy are applied only to the subset of data that pass the quality control thresholds at all previous levels. The first filters applied are the individual-level filters to remove entire individuals deviating excessively from the average across-individual coverage and error rate. Contig-level filters, followed by site-level filters, are then applied to remove entire contigs and sites, respectively, that appeared to be quality outliers. All individual specimens, contigs and sites should be filtered on multiple aspects of quality (e.g. potential cross-sample DNA contamination, sequencing errors, paralogy).

1. Filtering at individual level

a. Remove individuals having extremely low or high coverage. Individual coverage can be estimated using *8-ExonCaptureEvaluation Evaluation.* The file you want to examine is "~/Desktop/SeqCap/data/ExonCapEval/data_metrics.txt"

```
ke@NGS:~/Desktop/SeqCap/data$ less –S ExonCapEval/data_metrics.txt
```

b. Remove individuals with excessively high sequencing error rates measured as the percentage of mismatched bases out of the total number of aligned bases in the mitochondrial genome. Empirical error can be estimated using *6-AssemblyEvaluation COVERAGE*

To run *6-AssemblyEvaluation COVERAGE* you need first to generate pileup files for mitochondrial locus for each sample.

```
ke@NGS:~/Desktop/SeqCap/data$ 6-AssemblyEvaluation COVERAGE
```

```
Usage 6-AssemblyEvaluation COVERAGE [options]

Options:
-p   DIR        folder containing all pileup
                files generated by "samtools
               mpileup -f ref.fa sample1.bam
                > sample1.pileup"
-c   INT        coverage cutoff [5]
-q   INT        base quality cutoff [13]
```

2. Filtering at contig level

a. Remove contigs that show extremely low or high coverage based on the empirical coverage distribution across all contigs. *9-preFiltering* *percentile* can be used to generate a list of contigs that show extreme coverage based on percentile values (for example: 1% and 99%; 5% and 95% etc.). This list can then be used as one of input files in *10-SNPcleaner* for the purpose of filtering.

b. Remove contigs with at least one SNP having allele frequencies highly deviating from Hardy–Weinberg equilibrium expectations. Done by *10-SNPcleaner*. Note this is a very stringent filter even for exon capture dataset and not suitable at all for genomic dataset. To use this filter you need to provide "~/Desktop/SeqCap/data/bed_files/All_contig.bed" generated by *9-preFiltering* *bed.*

3. Filtering at site level

a. Remove sites with excessively low or high coverage based on the empirical coverage distribution. To determine high (e.g. 99% or 95%) and low (e.g. 1% or 5%) percentiles of base coverage you need run *9-preFiltering* *percentile* to get "CGRL_site_depth_percentile.txt".

b. Remove sites having allele frequencies highly deviating from Hardy–Weinberg equilibrium expectations (exact test). Done by *10-SNPcleaner*. This filter can be combined with the contig HWE filter (2.b).

c. Remove sites with biases associated with reference and alternative allele Phred quality, mapping quality and distance of alleles from the ends of reads. Also remove sites that show a bias towards SNPs coming from the forward or reverse strand. These will be done by *10-SNPcleaner*.

-> Strand Bias: Tests if variant bases tend to come from one strand.
-> End Distance Bias: Tests if variant bases tend to occur at a fixed distance from the end of reads, which is usually an indication of misalignment.
-> Base Quality Bias: Tests if variant bases tend to occur with a Phred-scale quality bias.
-> Mapping Quality Bias: Tests if variant bases tend to occur with a mapping quality bias.

d. Remove sites for which there are not at least M of the individuals sequenced at N coverage each. This makes sure that the remaining data matrix does not contain too much missing data. This will be done by *10-SNPcleaner*.

e. Remove sites with a root mean square (RMS) mapping quality for SNPs across all samples below a certain threshold. It is a measure of the variance of quality scores. This will be done by *10-SNPcleaner*.

f. (optional) For historic samples, characterize the pattern of base mis-incorporation first. Sometimes it is necessary to remove C to T and G to A SNPs from the dataset. This can be done by *10-SNPcleaner*.

Before running *10-SNPcleaner* make sure that individual-level filtering is finished.

Input:
1. "~/Desktop/SeqCap/data/pre-filtering/CGRL_gene_outside_percentile.txt" by *9-preFiltering* *percentile*.

2. "~/Desktop/SeqCap/data/bed_files/All_contig.bed" generated by *9-preFiltering* *bed*.

3. "~/Desktop/SeqCap/data/pre-filtering/CGRL_site_depth_percentile.txt" by *9-preFiltering* *percentile* is ready and will be used to guide the site-level coverage filtering.

4. Create a new folder "SNPcleaning" under "~/Desktop/SeqCap/data/" and inside this folder generate a raw vcf that contains all sites from all individual samples that pass individual-level filters:

```
ke@NGS:~/Desktop/SeqCap/data$ mkdir SNPcleaning
ke@NGS:~/Desktop/SeqCap/data$ cd SNPcleaning/
ke@NGS:~/Desktop/SeqCap/data/SNPcleaning$ samtools mpileup -B -D -I -S  -uf
~/Desktop/SeqCap/data/reference/in_target.fasta
~/Desktop/SeqCap/data/alignment/*sorted.bam | bcftools view -cg - > raw.vcf
```

-D	output per-sample DP in BCF
-B	disable BAQ computation
-I	do not perform indel calling
-S	output per-sample strand bias P-value in BCF

Commands:
#Run *10-SNPcleaner* under "~/Desktop/SeqCap/data/SNPcleaning/":

```
ke@NGS:~/Desktop/SeqCap/data/SNPcleaning$ 10-SNPcleaner -d 2 -D 7 -k 2 -u 1 -a 0
-B CGRL.bed -p CGRL_filtered -r ~/Desktop/SeqCap/data/pre-
filtering/CGRL_gene_outside_percentile.txt -X
~/Desktop/SeqCap/data/bed_files/All_contig.bed -g -v raw.vcf> out.vcf
```

Note: for “-D 7”, 7 is the 99% percentile of the base coverage. We get this number from “~/Desktop/SeqCap/data/pre-filtering/CGRL_site_depth_percentile.txt”.

Output:
In “~/Desktop/SeqCap/data/SNPcleaning/”, several files are created:

1. “CGRL.bed” contains sites (potentially variable and non-variable) passing all filters.

#To generate a keep file for ANGSD:
ke@NGS:~/Desktop/SeqCap/data/SNPcleaning$ cut -f1,2 CGRL.bed > CGRL.keep

2. “CGRL_filtered” (dumped with option -p) contains all sites that failed to pass certain filters. Characters in front of filtered sites indicate filters that the site failed to pass.

#To view this file:
ke@NGS:~/Desktop/SeqCap/data/SNPcleaning$ bunzip2 -c CGRL_filtered | less -S

3. “out.vcf” is the resulting vcf that contains sites (both variable and non-variable) passed all filters

Questions:
1. Check how many sites are present before and after filtering?
2. Check why some sites are filtered out by examining “CGRL_filtered”.