Design and usage of exon-capture-phylo: multi-sample protein-guided target recovery of orthologous exons for phylogenetics

Benjamin Y.H. Bai*

Moritz Lab, Research School of Biology, Australian National University.

Abstract

Exon capture is a cost-effective method for procuring thousands of orthologous loci from multiple samples for phylogenetic analysis. However, studies on novel or highly diverged sample organisms are often hampered by the lack of a suitable reference genome for target selection and/or guidance during target recovery. We present exon-capture-phylo: a sample-parallel recovery pipeline for captures where targets have been selected from annotated transcriptomes. The protein-guided algorithm allows the use of deeply diverged references. We processed four sample libraries from exon capture on Australian two-spined rainbow-skink (*Carlia amax*), guided by the reference proteome of the model reptile *Anolis carolinensis*. Despite an estimated sample-to-reference divergence of 150-200 Ma, 79.7% of the 3320 target exons were recovered for all samples.

*__Contact__: u5205339@anu.edu.au **Dates**: Nov 2013-Feb 2014 **Supervisor**: Dr. Jason Bragg **Course**: SCNC1102 - ASC2: Computational workflows for phylogenomics.

Contents

Part I
Design and Implementation

1 Introduction

Sequence capture techniques allow for the enrichment of target genomic sequences on the scale of entire exomes [1]. Probes synthesised to tile regions of interest are used to isolate complementary DNA fragments from a library preparation for amplification before sequencing. Applications include the selective re-sequencing of human genes to identify rare disease-associated variants. Hodges *et al.* [2] targeted 200K reference human coding regions, recovering up to 98% of intended targets.

In the phylogenetic context, target enrichment provides a cost-effective method for capturing informative orthologous loci from multiple samples across an unresolved taxonomic group [3]. Alignment and statistical analysis of recovered target exons can lead to inference of evolutionary relationships between sample taxa [4] without the steep costs associated with whole genome sequencing.

Where a reference genome is available [2, 5], target recovery often involves assembling captured reads that map directly to the reference at target locations. For studies on non-model organisms, there may be no suitable reference available for either mapping or target selection. Furthermore, when performing broad phylogenetic studies over highly diverged lineages, sequence divergence between samples may be so great that there is no single genomic reference onto which efficient mapping can be conducted.

To overcome these limitations, we introduce exon-capture-phylo: an automated protein-guided target recovery pipeline for exon capture data in non-model species, where capture targets are selected from *de novo* transcriptome assemblies from related taxa [6]. Transcriptomes are annotated using reciprocal best-hit BLASTx [7] to a reference proteome, followed by selection of target protein-coding sequences from the annotated loci. Each target is recovered via assembly of sample reads that align to its orthologous 'target' reference protein. As primary structures are generally more conserved than their protein-coding sequences, efficient alignment and recovery can be achieved even when the references used are very distantly related to the sample species. Source code is hosted on GitHub at `https://github.com/digitase/exon-capture-phylo`.

2 Methods

2.1 Pipeline organisation

exon-capture-phylo consists of a series of Perl scripts[1] that are linked with shell wrapper scripts, and organised into four broad phases: data preparation, exon assembly, variant calling, and output collation. It accepts three main datasets: target exon sequences selected from an annotated transcriptome, the reference proteome used during annotation, and captured sequence reads from each sample. Pipeline options are specified in a Bash-style configuration file.

In the data preparation phase, the configuration file is processed, and BLASTx databases are constructed. Each target is then recovered through *de novo* assembly of reads from each sample that align to the orthologous reference protein with BLASTx. SNP positions are called and incorporated into recovered target sequences as IUPAC nucleotide ambiguity codes. After all samples are completed, a gathering script collates, for each target, the recovered contigs from all samples; creating files ready for multiple sequence alignment.

The exon assembly and variant calling stages are data-parallel at the sample level. Sun Grid Engine (SGE) users can distribute samples to different computing nodes using SGE job arrays—equivalent to submitting a separate job for each sample simultaneously (`http://wiki.gridengine.info/wiki/index.php/Simple-Job-Array-Howto`). Non-SGE users can choose a number of samples to run in parallel, although these samples will run on a single machine only.

[1] Original scripts and workflow developed by Dr Jason Bragg, Moritz Lab, College of Medicine, Biology and Environment, Australian National University.

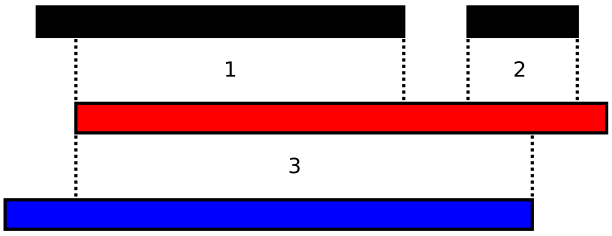

Figure 1: Length filtering process, showing target exon (blue), orthologous target protein (red), and two recovered assemblies (black), aligned using exonerate `--model protein2genome` (`http://www.ebi.ac.uk/~guy/exonerate/beginner.html`). Filtering requires the ratio of contig-to-protein overlap (regions 1 and 2 for the two contigs respectively) to exon-to-protein overlap (region 3) to exceed a given threshold (0.65 by default). In this case, only contig 1 satisfies the required criteria of $length(region1)/length(region3) \geq 0.65$. Contig 2 is too short, and is discarded.

2.2 Algorithmic description

Here we describe the target exon recovery process for one target exon, for one sample:

1. Sample reads are aligned to the orthologous target protein with BLASTx [8].
2. Aligned reads are assembled with Velvet [9] at a range of k-values (hash lengths).
3. CAP3 [10] is used to merge assemblies into less redundant contigs, requiring a 20 bp overlap with 99% sequence identity in the overlapping region.
4. Contigs are filtered for length (Fig. 1). Assemblies are aligned against the target protein with exonerate [11]. The target exon is also aligned to the target protein. The overlap proportion between the two overlapping ranges (measured in amino acids of the target protein) must exceed a threshold (0.65 by default) in order for a contig to pass filtering. This helps exclude assemblies of short but highly conserved sequence motifs that align with high bit-scores [12] to the target, yet are phylogenetically uninformative.
5. Filtered contigs are aligned to all proteins provided for the reference organism with BLASTx. The contig that aligned with the highest bit-score is selected as the 'best' contig if and only if the alignment subject is the protein orthologous to the target exon. This guards against the recovery of loci present in the reference that are paralogous rather than orthologous to the target exon, which would cause incorrect estimation of evolutionary rates during downstream phylogenetics [7].
6. Assuming a best contig was recovered, sample reads are mapped onto it with Bowtie 2 [13].
7. GATK variant calling pipeline [14]. Indels and SNPs are called with HaplotypeCaller, phased with ReadBackedPhasing, annotated, and filtered for support based on read depth. Well-supported SNP positions are written to the best contig as IUPAC nucleotide ambiguity codes.

2.3 Testing

De novo assembled transcriptomes for the *Carlia*, *Saproscincus*, and *Lampropholis* skink genera were annotated by reciprocal best-hit BLAST with the Ensembl proteome of the model reptile *Anolis carolinensis* (`ftp://ftp.ensembl.org/pub/release-67/fasta/anolis_carolinensis/pep/`). 3320 protein-coding sequences were chosen as target regions for exon capture from Australian reptile specimens using the NimbleGen SeqCap workflow (`http://www.nimblegen.com/products/seqcap/ez/developer/index.html`). Captured reads were cleaned with custom scripts; four sample libraries from two-spined rainbow skink (*Carlia amax*) specimens sampled over the Northern Territory were selected for analysis. Samples were run on parallel compute nodes as an SGE job array, using eight cores for BLASTx and 4 GB Java heap space per node. Peak memory usage was 6.8 GB per node; runtimes were 4-8 hours per sample, depending on cluster load and sample read count. Velvet k-values used were 31, 41, 51, and 61; filtering overlap threshold was 0.65.

Table 1: Summary of target recovery for each sample. Up to one best contig is selected as the representative sequence for each target exon per sample.

Metric	Sample 1	Sample 2	Sample 3	Sample 4	Total
Cleaned read count (M)	4.40	4.96	7.36	2.69	19.40
Short contigs count (excluded)	31539	28109	38331	30339	128318
Filtered contigs count	7527	7296	8075	7292	30190
Best contigs count	2921	2936	2804	2879	11540
Target count	3320	3320	3320	3320	13280
Recovery proportion	**0.880**	**0.884**	**0.845**	**0.867**	**0.869**

Table 2: Frequency tally of the number of contigs passing filtering for each target exon. A small non-zero number of contigs obtained per target suggests allelic assemblies; higher numbers ($\geq$4) of assemblies suggest that undesirable paralogous loci have been assembled.

Filtered contig count	Sample 1	Sample 2	Sample 3	Sample 4	Total
0	111	91	214	164	580
1	2248	2538	2178	2193	9157
2	687	475	643	681	2486
3	164	117	153	173	607
$\geq$4	110	99	132	109	450
Total	**3320**	**3320**	**3320**	**3320**	**13280**

3 Results and Discussion

Of the possible 13280 targets (four samples, 3320 exons per sample), 11540 were recovered, representing a recovery rate of 86.9% (Table 1). This is a reasonable recovery rate even though *A. carolinensis* is distantly related (approx. 150-200 Ma divergence) to our *C. amax* samples [15]. The ability to use such a distant protein reference expands the possible range of studiable organisms and phylogenetic scales. Using a reference diverged from all samples also reduces differential read-reference alignment efficiencies, minimising recovery biases that may occur if some samples were closely related to the reference.

Due to the large quantity of targets, we were able to impose relatively strict conditions to optimise the quality of our recovered assemblies. Velvet assemblies were done over many k-values to account for varying sample coverage [9], and CAP3 was used to merge Velvet assemblies only at essentially identical sequences. Over the 3320 target exons for each of our four samples, we assembled 158508 contigs using protein-aligned sample reads. A significant proportion of these were short assemblies, as only 19.0% passed the length filtering stage (Table 1), and the mean overlap for failed assemblies was 15%.

Table 2 summarises the distribution of filtered contigs amongst the samples and targets. In 3.4% of recoveries, four or more long contigs with no significant redundancy were recovered for a single target, suggesting the assembly of paralogous loci. Inclusion of paralogs in downstream phylogenetics is problematic, as their histories may not reflect the histories of their species [16]. By BLASTing candidate assemblies to the entire reference proteome, we test for recovery of paralogs not only within the target set, but across all known loci. A best contig for a target is selected if and only if it holds the highest bit score for *any* hit in the alignment, *and* this score occurred in an alignment to the orthologous reference protein. A total of 1096 highest-scoring candidates were rejected at this stage for incorrect alignment subject. Despite the stringent conditions, 79.7% of target exons were successfully recovered over all four samples (Table 3).

As the pipeline is dependent on the performance of existing bioinformatics software, opportunities for algorithmic optimisation were limited. Although alternative sequence aligners such as BLAT [17], or even dedicated mapping software are a faster choice for short read data, we favoured the BLAST group of algorithms for sensitivity advantages when aligning diverged sequences (`http://genome.ucsc.edu/FAQ/FAQblat.html`). A major bottleneck during the initial read-to-protein BLASTx can be mitigated with GNU parallel [18], which allows users to improve execution time by distributing segments of sample

Table 3: Tally of the number of samples out of four for which a best assembled contig was successfully recovered over 3320 target exons.

No. samples recovered for target	**0**	**1**	**2**	**3**	**4**	**Total**
Tally	299	60	50	264	2647	3320
Proportion	0.090	0.018	0.015	0.080	0.797	1.000

read files to many computational processes.

To adapt the pipeline for a wider range of studies, automatically-generated visualisations of the statistical information presented in the tables above—as well as read coverage and contig count on a per loci basis—would allow users to evaluate the impacts of parameter choice on target recovery for their particular datasets. The pipeline is currently restricted to Unix-based environments; to increase future accessibility, the workflow may be introduced onto a web-based platform such as Galaxy (http://galaxyproject.org/).

Part II
Usage

For version 0.1.0. (Feb 2014)

4 Installation

To install into the current directory: `git clone git://github.com/digitase/exon-capture-phylo.git`

5 Dependencies

exon-capture-phylo requires a Unix-based environment and is designed for the Bash shell.

Perl 5 http://www.perl.org/

makeblastdb ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

NCBI blastall ftp://ftp.ncbi.nlm.nih.gov/blast/executables/release/2.2.21/

> **Note:** `blastx` from the NCBI BLAST+ package is not yet supported due to performance concerns with short reads. Although performance improvements are reported for long queries and chromosome length databases [19], our timings revealed that `blastx` may be up to 3-fold slower when operating on short reads.

GNU parallel http://www.gnu.org/software/parallel/

Velvet https://www.ebi.ac.uk/~zerbino/velvet/

CAP3 Sequence Assembly Program http://seq.cs.iastate.edu/cap3.html

exonerate http://www.ebi.ac.uk/~guy/exonerate/

bowtie2 http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

samtools http://samtools.sourceforge.net/

Java 7 http://java.com/en/

Picard http://picard.sourceforge.net/index.shtml

Genome Analysis Toolkit (GATK) http://www.broadinstitute.org/gatk/

BioPerl http://www.bioperl.org/

6 Input datasets

In the following section, a FASTA 'record' consists of a 'sequence ID' on lines beginning with '>', followed on the next line by the 'sequence', which may span multiple lines.

6.1 Target protein sequences

The pipeline was designed to process Ensembl peptide references.

- Text file listing the sequence IDs for each target protein, one ID per line.
  - Sequence IDs should be alphanumeric, *avoiding* the use of punctuation or whitespace.

    e.g. `ENSACAP00000015077`
- Peptide FASTA file containing at least the records for each target protein listed in the text file.
  - Additional records will be ignored. The sequence ID for each sequence must be the first whitespace-delimited component of the line, and identical to the corresponding sequence ID in text file..

    e.g. `ENSACAP00000015721` is the first component of the correctly formatted sequence ID line:

    `>ENSACAP00000015077 pep:novel scaffold:AnoCar2.0:GL343198.1:3158633:3179784:-1`

6.2 Target exon sequences

- Text file listing the sequence IDs for each target exon, one ID per line.
  - The format of the ID should be `<orthologous_protein_ID>_<rest_of_exon_name>`
  - The orthologous protein ID should *avoid* the use of of punctuation or whitespace, as specified in Section 6.1, and it must occur directly before the *first* separating underscore in the line.

    e.g. `ENSACAP00000015077_exon1_Carlia` represents a target exon orthologous to the `ENSACAP00000015077` reference protein.
- Nucleotide FASTA file containing at least the records for each target exon listed in the text file.
  - Additional records will be ignored. The sequence ID for each sequence must be identical to the corresponding sequence ID listed in the text file.

6.3 Sample reads

- Text file listing the samples names, one name per line.
  - Sample names should contain no whitespace.

    e.g. `SP04_indexing10`
- Three Illumina FASTQ paired-end read files are required per sample listed above, representing forward, reverse and unpaired captured reads.
  - Reads should be pre-cleaned (quality/length filtering, adapter trimming etc.) and compressed with `gzip`. The three files for each sample should be respectively named:

    `<sample_name>_<FORWARD_READS_SUFFIX>.fastq.gz`

    `<sample_name>_<REVERSE_READS_SUFFIX>.fastq.gz`

    `<sample_name>_<UNPAIRED_READS_SUFFIX>.fastq.gz`
  - Filename suffices are specified in the configuration file (Section 7.1).

    e.g. `SP04_indexing10_1_final.fastq.gz` has `FORWARD_READS_SUFFIX="1_final"`

7 Configuration

7.1 Configuration file

The configuration file is a text document placed in the same directory as `exon-capture-phylo.sh` that declares variables regarding input/output files and directories, performance, algorithmic considerations, and dependency locations in Bash assignment syntax.

- Parameters should be listed as `PARAMETER_NAME="value"`
- All pathnames should be absolute (required when using SGE). Directory paths should be terminated with '`/`'.
- The use of double quotes around parameter values that contain whitespace is essential. For safety, all non-trivial parameter values should be quoted.
- Lines beginning with '`#`' are ignored.

Recognised parameters are listed below with example value formats. Avoid using parameters not listed below, as the configuration file will be sourced as a valid bash script.

General

`SCRIPT_DIR` Directory containing the main script `exon-capture-phylo.sh` and *this config file*.

e.g. `SCRIPT_DIR="~/project/exon-capture-phylo/"`

`OUT_DIR` Directory for pipeline output.

e.g. `OUT_DIR="~/project/ecp_output/"`

Dataset Input formats are specified in Section 6.

`SAMPLES_LIST` Text file listing sample names.

`SAMPLES_DIR` Directory containing cleaned sample read files (forward, reverse and unpaired).

`FORWARD_READS_SUFFIX, REVERSE_READS_SUFFIX, UNPAIRED_READS_SUFFIX` Filename suffixes such that, for example, `<sample_name>_<FORWARD_READS_SUFFIX>.fastq.gz` is the forwards read file located in `SAMPLES_DIR` for the sample named `<sample_name>`. Also see Section 6.3.

`TARGET_PROTEIN_SEQS_LIST` Text file listing target protein IDs.

`ALL_PROTEIN_SEQS` Protein FASTA file containing at least the protein records listed above.

`TARGET_EXON_SEQS_LIST` Text file listing target exon IDs.

`ALL_EXON_SEQS` Nucleotide FASTA file containing at least the exon records listed above.

Resource

`XARGS_PARALLEL_SAMPLES` When *not* using SGE job arrays, this is the number of samples processed in parallel. Comparable to the SGE job array option `-tc` (Section 7.2). Ignored when using SGE job arrays.

e.g. `XARGS_PARALLEL_SAMPLES=2`

`BLAST_PROCS_PER_SAMPLE` Number of processes used per sample when BLASTing sample reads against target proteins. When using SGE job arrays, this is the number of processes used by BLASTx on each compute node.

e.g. `BLAST_PROCS_PER_SAMPLE=8`

`JAVA_MAX_HEAP_SIZE` Amount of heap memory (gigabytes) per sample provided to the Java virtual machine during variant calling (Picard and GATK). The required memory varies with sample read counts. Increasing heap size may provide a small performance boost, subject to the law of diminishing returns. See Section 10.3 for memory troubleshooting.

e.g. `JAVA_MAX_HEAP_SIZE=4`

Assembly See Section 10.4 for recommendations.

`BLAST_EVALUE` Expectation value used for initial BLASTx of sample reads against target proteins. The smaller the value, the more specific and less sensitive the alignment will be, thus recovering fewer reads per target.

e.g. `BLAST_EVALUE="1e-9"`

`VELVET_K_VALUES` K-mer lengths used during Velvet assemblies. Must be odd, and smaller than the read length. A good spread is recommended to account for coverage variations.

e.g. `VELVET_K_VALUES=(31 41 51 61)`

`MIN_OVERLAP` Minimum length proportion that an assembled contig-to-target protein alignment must coincide with the corresponding target exon-to-target protein alignment for that contig to pass the exonerate filtering phase (Fig. 1).

e.g. `MIN_OVERLAP="0.65"`

Dependencies For pre-installed programs included in the `$PATH` environment variable, the program name is sufficient.

`BLASTALL_PATH` Path to `blastall` binary.

e.g. `BLASTALL_PATH="blastall"`

`MAKEBLASTDB_PATH` Path to `makeblastdb` binary.

`VELVETH_PATH` Path to `velveth` binary.

`VELVETG_PATH` Path to `velvetg` binary.

`EXONERATE_PATH` Path to `exonerate` binary.

`CAP3_PATH` Path to `cap3` binary.

e.g. `CAP3_PATH="~/software/CAP3/cap3"`

`BOWTIE2_BUILD_PATH` Path to `bowtie2-build` binary.

`BOWTIE2_PATH` Path to `bowtie2` binary.

`SAMTOOLS_PATH` Path to `samtools` binary.

`PICARD_DIR` Picard directory path.

e.g. `PICARD_DIR="~/software/picard-tools-1.104/"`

`GATK_DIR` GATK directory path.

e.g. `GATK_DIR="~/software/GenomeAnalysisTK-2.6-4-g3e 5ff60/"`

7.2 SGE configuration

To use SGE job arrays, SGE configuration options must be placed at the beginning of the main script `exon-capture-phylo.sh` before any code lines. Configuration lines must begin with '`#$`'. Options are ignored when not using SGE.

General

`-cwd` Run job from current directory. Causes job info and error message files (Section 10.1) to appear in the current directory rather than your home directory.

`-M <email>` Email address to receive SGE job information emails. Contains information on host, runtime, job exit status, and memory usage.

`-m bea` Send email to above address at job beginning, end, and in case of job abortion.

`-r y` Rerun job if aborted i.e. in the case of a compute node crash.

`-N <job_name>` Set the job name (`$JOB_NAME` environment variable) to `job_name`.

Resource

`-l virtual_free=<n1>G,h_vmem=<n2>G` Request `n1` GB physical and virtual memory total, and terminate job if memory usage exceeds `n2` GB.

- `virtual_free` must be higher than `JAVA_MAX_HEAP_SIZE` to account for non-GATK memory usage during variant calling.
- `h_vmem` should be higher than `virtual_free`; it represents the maximum memory use to be tolerated.

Note: Some SGE environments use the (roughly) equivalent resource `mem_free`. For more information on which resources are requestable on your SGE environment, use `qconf -sc`, or contact the cluster administrators.

`-tc <n>` Throttle the maximum number of samples run in parallel i.e. use up to `n` compute nodes in parallel. Comparable to the non-SGE option `XARGS_PARALLEL_SAMPLES`.

`-t 1-<n>` Schedule `n` array jobs, where `n` equals the number of samples listed in `SAMPLES_LIST`.

For more configuration options, refer to the Grid Engine User Commands manual: `man qsub`

8 Execution

When using SGE job arrays, `cd` to `SCRIPT_DIR`, then use `qsub exon-capture-phylo.sh <config_file>` to submit to queueing system. When *not* using SGE: `./exon-capture-phylo.sh <config_file>`

- *Ensure* that `<config_file>` is placed in `SCRIPT_DIR`.
- *Do not* modify or move the config file during execution.
- When using SGE job arrays, in addition to a valid config file, `exon-capture-phylo.sh` must be configured according to Section 7.2.

8.1 Reducing I/O demands

Due to the potentially large number of target exons being processed in parallel, the pipeline is relatively demanding in terms of I/O performance. At the cost of pipeline runtime, short sleep periods between processing each exon can be introduced to lighten this demand by uncommenting `sleep(2)` lines in `bestcontig_distrib.pl`, `callVelvetAssemblies.pl` and `catcontigs.pl`

9 Output

Pipeline output occurs in `OUT_DIR`. Main outputs include the recovered targets, mapping coverage, and variant information. A summary of all intermediate pipeline files can be found in Section 12.1.

9.1 Assemblies

These directories contain the pipeline's main outputs: recovered targets as assembled contig(s) from each sample for each target exon.

`OUT_DIR/gathercontigs/`

`all_contigs/` Contains a nucleotide FASTA file for each target exon, with records for all assembled contigs for that target exon that passed length filtering. Source sample name is indicated in the sequence ID.

`countallcontigs.txt` A text file with tab-delimited fields that indicate the number of assembled contigs that passed length filtering for each target exon (rows), for each sample (columns).

`best_contigs/` Contains a nucleotide FASTA file for each target exon. Each file contains up to one 'best' contig (as defined in Section 2.2) for each sample with the sample name as the sequence ID, if such a best contig exists.

OUT_DIR/gatherAmbigcontigs/ Contains the same FASTA files as `best_contigs/`, except SNP positions discovered during variant calling are represented by IUPAC nucleotide ambiguity codes. These are the final recovered targets.

9.2 Coverage statistics

Mapping coverage during the variant-calling stage is analysed with GATK `DepthOfCoverage.jar`, with output directory `OUT_DIR/<sample_name>/<sample_name>_gatkSNPcalls/`

- `<sample_name>.ReadGrouped.DepthOfCoverageTable`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_cumulative_coverage_counts`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_cumulative_coverage_proportions`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_interval_statistics`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_interval_summary`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_statistics`
- `<sample_name>.ReadGrouped.DepthOfCoverageTable.sample_summary`

File content formats are are described at http://www.broadinstitute.org/gatk/gatkdocs/org_broadinstitute_sting_gatk_walkers_coverage_DepthOfCoverage.html

9.3 Filtered variant counts

For each sample, contains summary of the variants discovered in each recovered target. Tallies number of filtered heterozygous, homozygous alternate, and indel sites called by GATK. Outputs in `OUT_DIR/<sample_name>_vcf2ambigfasta/<sample_name>_best2refs.vcf2ambigfasta_refs.stats`

10 Troubleshooting

10.1 Info and error messages

The pipeline produces info and error message streams for each sample analysed:

Info messages Contains timestamps indicating the start and end of each pipeline segment.

- When using SGE job arrays, messages for each sample appear in `SCRIPT_DIR/<job_name>.o<qsub_job_ID>.<SGE_TASK_ID>`, where `qsub_job_ID` is a number assigned to each queued job by SGE, and `SGE_TASK_ID` represents the 1-based index of the sample name in `SAMPLE_LIST`.
- When *not* using SGE job arrays, info messages appear in the standard output stream.

Error messages and core dumps Messages that may require user attention.

- When using SGE job arrays, messages for each sample appear in `SCRIPT_DIR/<job_name>.e<job_number>.<SGE_TASK_ID>`
- When *not* using SGE job arrays, error messages appear in the standard error stream.
- Possible messages include:
  - `"No velvet assembled contigs for <target_protein>"` indicates Velvet failed to produce any assemblies using the sample reads that aligned to the target protein. CAP3's behaviour on empty input is a segmentation fault, producing a `core.<random_number>` memory dump file in `OUT_DIR`. These core dump files serve no purpose and can be safely removed. A missing CAP3 output then triggers errors from cat (`No such file or directory`) and exonerate (`** FATAL ERROR **: No sequences found`), however this does not affect the pipeline downstream, besides failure to recover that particular target for the sample.

  - `"... failed filtering"` indicates assembled contigs that failed to pass filtering. Required and achieved overlap proportions are displayed.
  - `"Rejected the highest-scoring contig ..."` indicates target exons for which there was no best contig selected, as the highest-scoring contig aligned to a non-orthologous loci and was rejected (Section 2.2).

10.2 Temporary files

When pipeline execution is interrupted before completion, and SGE job arrays are in use, the following temporary files may remain in `OUT_DIR`.

- `preparing_data.lock`
- `prepare_data_complete.lock`
- `completed_ids.txt`
- `gather_data.lock`

These files are involved in coordination of parallel samples, and may be safely removed. For more information, see Section 12.2.

10.3 Memory usage errors

The following error messages usually appear when the amount of memory required or requested by the pipeline exceeds the amount of memory available, or requested from SGE:

- `Error occurred during initialization of VM`
- `Could not reserve enough space for object heap`
- `Error: Could not create the Java Virtual Machine.`
- `Error: A fatal exception has occurred. Program will exit.`

When using SGE, ensure that the value of `virtual_free` requested in the `-l` SGE parameter is at least 2-3 GB higher than `JAVA_MAX_HEAP_SIZE` to account for script memory usage during GATK. When *not* using SGE, check that the amount of physical memory available is 2-3 GB higher than `JAVA_MAX_HEAP_SIZE`.

- The converse error is `OutOfMemoryError`, where the amount of memory required to process the sample is not available, and `JAVA_MAX_HEAP_SIZE` should be increased.

10.4 Assembly parameter optimisation

Note: See Section 12.1 for description of diagnostic files.

If few targets are being recovered in `OUT_DIR/gatherAmbigcontigs/`:

Cause For runs with extreme sample-to-reference divergence, very few reads may align to the target protein at the selected `BLAST_EVALUE`.

Diagnostic Low number of reads for many samples in

```
OUT_DIR/<sample_name>/<target_protein>/
    <target_protein>_call_velvet_assemblies/<target_protein>_all_hitreads.fasta
```

Solution Increase `BLAST_EVALUE` to increase alignment sensitivity at the cost of specificity. For more information on the role of BLAST expectation values, see http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=FAQ#expect.

Cause Poor selection of `VELVET_K_VALUES` for sample coverages.

Diagnostic Low number of contigs for many samples in

`OUT_DIR/<sample_name>/<target_protein>/`

`<target_protein>_catcontigs/<target_protein>_velvet_contigs.fasta`

Solution Increase range of `VELVET_K_VALUES`, which should not have a large effect on runtime as the assembly process is not expensive. For more information on k-value choice and the effects of read coverage, see `http://www.ebi.ac.uk/~zerbino/velvet/hash_length_choice.html` and the Velvet manual [20]. Also try increasing `BLAST_EVALUE` to involve more reads in the assembly process.

Cause `MIN_OVERLAP` value too stringent. The majority of assembled contigs are short.

Diagnostic Low number of filtered contigs for many samples in

`OUT_DIR/<sample_name>/<target_protein>/`

`<target_protein>_bestcontig_distrib/`

`<target_exon>_velvet_contigs.cap3ed.exonerated.filtered.fasta`

Solution `Reduce MIN_OVERLAP`. Also try increasing `BLAST_EVALUE` to involve more reads in the assembly process.

Cause Paralogous loci being assembled and rejected.

Diagnostic High numbers of rejected highest-scoring loci in the error message file

`SCRIPT_DIR/<job_name>.e<job_number>.<SGE_TASK_ID>`

Solution Increase `MIN_OVERLAP` to filter out short but high-scoring loci. Also try decreasing `BLAST_EVALUE` to involve fewer reads in the assembly process.

11 Demo usage

Install with `git clone git://github.com/digitase/exon-capture-phylo.git`

Demo datasets are provided in `demo_data.tar.gz` (See GitHub `README.md` for link). These are reduced versions of the test datasets in Section 2.3, allowing recovery on five exon capture targets over two samples:

`demo_data/`

`target_proteins/`

`Anolis_carolinensis.AnoCar2.0.67.pep.all.fa` Ensemble peptide reference for the model reptile *Anolis carolinensis*.

`anolistargetproteins.txt.5` Five target protein IDs of the original 3226 selected protein targets.

`target_exons/`

`targetexons.fasta` Orthologous exons from the *Carlia*, *Saprosc*incus, and *Lampropholis* skink genera.

`targetexons.txt.5` The five target exon IDs to recover.

`samples/`

`trunc100000_SP04_indexing<10|11>_<1|2|u>_final.fastq.gz` Six cleaned sample read files captured from two samples of C*arlia amax*, each truncated to 100K reads.

`sample_names.txt` The two sample names.

The provided `demo.config` must be modified to reflect the absolute paths on your installation according to Section 7. Move the contig file to the same directory (`SCRIPT_DIR`) as `exon-capture-phylo.sh`, then `cd` to that directory.

- When using SGE job arrays, submit with `qsub exon-capture-phylo.sh demo.config`. The recovery will use 2 compute nodes in parallel, using 4 processes and approximately 6 GB peak memory on each node, with default resource parameters.
- When *not* running SGE job arrays, execute with `./exon-capture-phylo.sh demo.config`. The recovery will use 8 processes and approximately 12 GB peak memory to process both samples in parallel, with default resource parameters.

Note: Due to the low read counts, SNPs will have low support and be filtered out, thus the output at `OUT_DIR/gatherAmbigcontigs` may not differ from `OUT_DIR/gathercontigs`.

12 For developers

12.1 Comprehensive summary of intermediate files

These files are generated by the pipeline during intermediate assembly and variant-calling stages. They can be safely removed when all output useful for downstream analysis has been extracted from `OUT_DIR`. Undescribed files with a `.log` suffix contain runtime output redirected from dependencies.

`OUT_DIR/core.<number>` Core dumps from non-recovered targets. See Section 10.1.

`OUT_DIR/blast_dbs/`

`all_proteins.fasta` Reference proteome file `ALL_PROTEIN_SEQS` with sequence IDs trimmed to the first whitespace-delimited component.

`target_proteins.fasta` Target proteins file, containing those records from `all_proteins.fasta` with sequence IDs listed in `TARGET_PROTEIN_SEQS_LIST`.

`*.<|pin|psq|phr>` Protein BLAST database files generated from the corresponding FASTA.

`OUT_DIR/<sample_name>/`

`<sample_name>_<FORWARD_READS_SUFFIX|REVERSE_READS_SUFFIX|UNPAIRED_READS_SUFFIX>`

`.fasta` Records in the corresponding `.fastq.gz` reads file converted to FASTA format.

`.against_targets.blast` Tabular BLASTx output file from BLASTing above FASTA file against the `target_proteins` database.

`<target_protein>/`

`<target_protein>_assemble_by_prot/`

`<target_protein>_<1|2|u>_hitreads.fasta` Reads that aligned to the target protein from the forward, reverse and unpaired sample read files.

`<target_protein>_<1p|2p>_hitreads.fasta` Reads that pair to reads in the forward and reverse `_hitreads` files are also collected.

`<target_protein>_call_velvet_assemblies/`

`<target_protein>_k<k_value>/` Assembled contigs at that k-value (`contigs.fa`), reads used from all `_hitreads` files (`<target_protein>_all_hitreads.fasta`), and other Velvet output files described in the Velvet manual [20].

`<target_protein>_velvet_contigs.fasta` Collated assemblies from `contigs.fa` files from all k-values.

`<target_protein>_catcontigs/`

`<target_protein>_velvet_contigs.fasta` Copy of collated assemblies from `contigs.fa` files from all k-values.

`<target_protein>_velvet_contigs.fasta.cap.contigs` Merged contigs from CAP3 merging of collated assemblies.

<target_protein>_velvet_contigs.fasta.cap.singlets Non-redundant contigs not used in the CAP3 merging process. Other CAP3 output files are indicated with with `.cap.` as part of the filename, and are described in the documentation included with the CAP3 package (http://seq.cs.iastate.edu/cap3.html).

<target_protein>_velvet_contigs.cap3ed.fasta Collated CAP3 `_contigs` and `_singlets` file.

<target_protein>.fasta Target protein sequence, extracted from the reference.

<target_protein>_velvet_contigs.cap3ed.exonerated.fasta Exonerate alignment of collated CAP3 contigs to the target protein sequence. Alignment range is shown in each sequence ID.

<target_protein>_bestcontig_distrib/

<target_exon>.fasta Target exon sequence.

<target_exon>.exonerated.fasta Exonerate alignment of target exon sequence to the target protein sequence. Alignment range is shown in the sequence ID.

<target_exon>_velvet_contigs.cap3ed.exonerated.filtered.fasta Exonerated CAP3 contigs that passed length filtering, with a sufficient proportion of alignment range overlap to the target exon-protein alignment.

<target_exon>_velvet_contigs.cap3ed.exonerated.filtered.against_all.blast Tabular BLASTx output of alignment between filtered contigs and reference proteome database `all_proteins`.

<target_exon>_velvet_contigs.cap3ed.exonerated.filtered.best_contig.fasta Contig with the highest BLAST bit-score, if the contig aligned to the orthologous target protein.

OUT_DIR/<sample_name>/

<sample_name>_best2refs/

<sample_name>_best2refs.fasta Collation of the best contigs recovered for each target exon.

<sample_name>_best2refs.dict Picard `CreateSequenceDictionary.jar` dictionary. Used by GATK.

<sample_name>_best2refs.fasta.fai `samtools faidx` FASTA index file. Used by GATK.

<sample_name>_mapsnp/

***.bt2** bowtie2-build index files.

<sample_name>.sorted.bam Sorted BAM file from `bowtie2` mapping of sample reads onto the best contigs.

<sample_name>_gatkSNPcalls/

<sample_name>.ReadGrouped.bam Sorted BAM file with simple read group added by Picard `AddOrReplaceReadGroups.jar`

<sample_name>.ReadGrouped.bam.bai `samtools index` for read-grouped BAM file.

<sample_name>.ReadGrouped.*.vcf A series of VCF files generated during the GATK pipeline (see Section 2.2). Coverage statistic files that contain `DepthOfCoverageTable` in the filename are listed in Section 9.2.

<sample_name>.ReadGrouped.*.vcf.idx VCF index files. Used by GATK.

<sample_name>_vcf2ambigfasta/

<sample_name>_best2refs.vcf2ambigfasta_refs.fasta The best contig records with IUPAC ambiguity codes added from variant calling.

<sample_name>_best2refs.vcf2ambigfasta_refs.stats Information regarding the number and type of variants that passed filtering for each target exon. Described in Section 9.3.

12.2 Mutex files when running SGE job arrays

Since data preparation and gathering must be completed only once across all samples, the pipeline uses `mkdir` and mutual exclusion directories to coordinate which sample's array job executes these scripts.

`preparing_data.lock` When present, indicates the data preparation phase is in progress. Created by the first sample to reach the data preparation phase.

`prepare_data_complete.lock` When present, indicates the data preparation phase is complete. Created by the first sample to reach the data preparation phase, replacing `preparing_data.lock`.

`completed_ids.txt` List of `SGE_TASK_ID`s (samples) completed to the interrupted point. Updated when each sample completes the variant-calling phase.

`gather_data.lock` When present, indicates the final data collation phase is in progress. Created by the last sample to complete the variant-calling phase, when the number of completed samples in `completed_ids.txt` equals the sample number. All mutex files are removed when data gathering completes successfully.

12.3 Perl Plain Old Documentation

Each core Perl script (`.pl`) in the pipeline contains perlpod documentation (http://perldoc.perl.org/perlpod.html), viewable with `perldoc <script_name>.pl`. Documentation format is modelled on CPAN requirements.

13 Abbreviations

Ma Mega-annum: one million years

K thousand

BLAST Basic Local Alignment Search Tool

SNP single-nucleotide polymorphism

IUPAC International Union of Pure and Applied Chemistry

GATK The Genome Analysis Toolkit

SGE Sun Grid Engine *(a.k.a.* Oracle Grid Engine), a batch-queueing system for computing clusters.

bp base pairs

GB Gigabyte

M million

I/O input/output

BAM Sequence Alignment/Map format, binary version

VCF Variant Call Format

14 Acknowledgements

- Dr. Jason Bragg, project supervisor, for his expertise, enthusiasm, tireless mentoring, and the original Perl scripts that form the core of exon-capture-phylo;
- Professor Craig Moritz, project co-marker and lab leader, for his guidance on project direction;
- Professor Scott Keogh, project co-marker, for attending the project seminar;
- Administrators of the Genome Discovery Unit cluster, for the use of their computing resources;
- And the rest of the Moritz Lab and Moritz Lab Summer Scholars, for their friendliness and support.

References

[1] Bainbridge, M., Wang, M., Burgess, D., Kovar, C., Rodesch, M., *et al.* (2010). 'Whole exome capture in solution with 3 Gbp of data.' *Genome Biology* **11**(6):R62. URL http://genomebiology.com/2010/11/6/R62.

[2] Hodges, E., Xuan, Z., Balija, V., Kramer, M., Molla, M.N., *et al.* (2007). 'Genome-wide in situ exon capture for selective resequencing.' *Nature Genetics* **39**(12):1522–1527.

[3] Lemmon, E.M. and Lemmon, A.R. (2013). 'High-throughput genomic data in systematics and phylogenetics.' *Annual Review of Ecology, Evolution, and Systematics* **44**:99–121.

[4] Yang, Z. and Rannala, B. (2012). 'Molecular phylogenetics: principles and practice.' *Nature Reviews Genetics* **13**(5):303–314.

[5] Ng, S.B., Turner, E.H., Robertson, P.D., Flygare, S.D., Bigham, A.W., *et al.* (2009). 'Targeted capture and massively parallel sequencing of 12 human exomes.' *Nature* **461**(7261):272–276.

[6] Bi, K., Vanderpool, D., Singhal, S., Linderoth, T., Moritz, C., and Good, J.M. (2012). 'Transcriptome-based exon capture enables highly cost-effective comparative genomic data collection at moderate evolutionary scales.' *BMC Genomics* **13**(1):403.

[7] Wall, D., Fraser, H., and Hirsh, A. (2003). 'Detecting putative orthologs.' *Bioinformatics* **19**(13):1710–1711.

[8] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). 'Basic local alignment search tool.' *Journal of Molecular Biology* **215**(3):403–410.

[9] Zerbino, D.R. and Birney, E. (2008). 'Velvet: algorithms for de novo short read assembly using de Bruijn graphs.' *Genome Research* **18**(5):821–829.

[10] Huang, X. and Madan, A. (1999). 'CAP3: A DNA sequence assembly program.' *Genome Research* **9**(9):868–877.

[11] Slater, G.S. and Birney, E. (2005). 'Automated generation of heuristics for biological sequence comparison.' *BMC Bioinformatics* **6**(1):31.

[12] Madden, T. (2002). *Chapter 16: The BLAST Sequence Analysis Tool, The NCBI handbook*. National Center for Biotechnology Information (US).

[13] Langmead, B. and Salzberg, S.L. (2012). 'Fast gapped-read alignment with Bowtie 2.' *Nature Methods* **9**(4):357–359.

[14] DePristo, M.A., Banks, E., Poplin, R., Garimella, K.V., Maguire, J.R., *et al.* (2011). 'A framework for variation discovery and genotyping using next-generation DNA sequencing data.' *Nature Genetics* **43**(5):491–498.

[15] Hedges, B.S. and Vidal, N. (2009). *The Timetree of Life: Lizards, snakes, and amphisbaenians (Squamata)*. Oxford University Press.

[16] Fitch, W.M. (1970). 'Distinguishing homologous from analogous proteins.' *Systematic Biology* **19**(2):99–113.

[17] Kent, W.J. (2002). 'BLAT - the BLAST-like alignment tool.' *Genome Research* **12**(4):656–664.

[18] Tange, O. (2011). 'Gnu parallel-the command-line power tool.' *The USENIX Magazine* **36**(1):42–47.

[19] Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T.L. (2009). 'BLAST+: architecture and applications.' *BMC Bioinformatics* **10**(1):421.

[20] Zerbino, D. (2008). 'Velvet Manual-version 1.1.'