DR. MICHAEL G. CAMPANA (Orcid ID : 0000-0003-0461-6462)

Article type : Resource Article

BaitsTools: software for hybridization capture bait design

Michael G. Campana

Center for Conservation Genomics, Smithsonian Conservation Biology Institute, 3001 Connecticut Avenue NW, Washington, DC 20008, USA

Keywords: hybridization capture, bait, targeted sequencing, software, nucleic acid

Corresponding author: Michael G. Campana
Center for Conservation Genomics, Smithsonian Conservation Biology Institute, 3001 Connecticut Avenue NW, Washington, DC 20008, USA.
Fax: +1 202-633-1237; E-mail: campanam@si.edu

Running title: BaitsTools

Abstract

Nucleic acid hybridization capture is a principal technology in molecular ecology and genomics. Bait design, however, is a non-trivial task and few resources currently exist to automate the process. Here, I present BaitsTools, an open-source, user-friendly software package to facilitate the design of nucleic acid baits for hybridization capture.

Introduction

Targeted high-throughput sequencing using hybridization capture (e.g. Gnirke *et al*. 2009) is a critical tool in molecular ecology and genomics. Applications include genomic investigations of non-model organisms using ultra-conserved elements (Faircloth *et al*. 2012; Lim & Braun 2016), exome capture (e.g. Ng *et al*. 2009), single nucleotide polymorphism (SNP) analysis (e.g. Burbano *et al*. 2010), targeted metagenomics (e.g. Campana *et al*. 2016), and ancient DNA enrichment and museomics (e.g. Burbano *et al*. 2010; Hawkins *et al*. 2016; Lim & Braun 2016), among others. Hybridization capture utilizes oligonucleotide baits to enrich target molecules from nucleic acid libraries through hybridization of the baits to complementary nucleotide sequences in the libraries, isolation of the hybridized molecules, and removal of the non-target library molecules. Manual bait design is non-trivial, and few software packages are publicly available for this task (see, for instance, Faircloth 2017). Here, I describe BaitsTools, an open source package to design and *in silico* test bait sequences for a variety of hybridization capture applications.

BaitsTools functions

BaitsTools generates high-quality oligonucleotide baits from a variety of input formats using input-specific subcommands (Table 1). Subcommand parameters are user customizable with defaults suited for generating 120 bp RNA baits (such as MYbaits® from MYcroarray). Currently, BaitsTools can generate baits from FASTA/FASTQ sequences and alignments, Stacks (Catchen *et al*. 2011, 2013) population summary statistics files, genome annotations and features (BED/GTF/GFF), PyRAD and ipyrad loci files (Eaton 2014), and VCF files. The software can also analyze and filter previously generated bait sequences using the checkbaits subcommand. BaitsTools utilizes a three-step workflow: variant selection, bait generation, and bait quality control and filtration (Figure 1). Depending on the selected subcommand and user requirements, some of these steps can be omitted. BaitsTools can output detailed log files giving locus- and subcommand-specific results for each of these steps.

Variant selection

Genome sequencing and reduced-representation approaches (such as RADseq) often discover orders of magnitude more sequence variants than are typically analyzed in genomic projects using hybridization capture. BaitsTools can select variants from VCF, PyRAD and ipyrad LOCI files, and Stacks population summary statistics files to identify a subset of variants evenly spaced across genomes. Genome assemblies vary significantly in quality – ranging between a selection of assembled reduced-representation loci to contig-, scaffold- or chromosome-level whole-genome assemblies. Hereafter, individual component assembled sequences are referred to as 'contigs' for simplicity. To ensure even spacing across reference sequences of varying quality, the user can select a maximum number of variants per contig or can scale the number of selected variants per individual contig by its length. The first option is useful for highly fragmented assemblies or reduced-representation datasets without a genome assembly in order to sample as many genomic markers as possible, whereas the latter is appropriate for high-quality genome assemblies where most polymorphisms are located on the longest contigs. The user can also specify a minimum physical distance between selected variants to ensure equal coverage across the reference sequences and mitigate linkage disequilibrium. Additionally, stacks2baits can sort polymorphisms varying within or between populations and by deviation from Hardy-Weinberg equilibrium according to a χ^2 test. Finally, vcf2baits can exclude sequence variants below a minimum specified Phred-like quality score.

Selected variants are output in a new VCF or Stacks summary table for the vcf2baits and stacks2baits commands respectively. Furthermore, the BaitsTools variant-selection and bait-generation options are appended to the end of the VCF header for future user reference.

Bait generation

To generate candidate bait sequences, BaitsTools imports reference sequences in FASTA/FASTQ or PyRAD/ipyrad LOCI format. For the aln2baits and tilebaits commands, the input nucleotide alignments or sequence lists are treated as reference sequences. BaitsTools can also generate baits across the break in linearized circular sequences (e.g. complete mitogenomes in FASTA format). Appending '#circ' to the end of a sequence header indicates to BaitsTools that a sequence is circular. Otherwise, BaitsTools assumes linear sequences.

After reference sequence importation, baits are generated according to each subcommand's algorithm. For vcf2baits and stacks2baits, the regions surrounding the selected variant are extracted from the reference sequence using genomic coordinates. The extracted region is determined by the specified bait length, tiling density, and position of the selected variants within the candidate bait. Optionally, alternate alleles are then applied to the obtained bait sequences to produce a balanced bait set representing all known alleles equally.

The tilebaits subcommand divides the imported sequences into baits based on requested bait length and tiling density. The annot2baits and bed2baits subcommands extract specified genomic features from the reference sequences. The extracted sequences are output in FASTA format for user reference. The extracted sequences are then passed to tilebaits to generate the candidate baits. Similarly, the aln2baits divides the alignment into windows based on desired bait length and tiling density. Baits are then generated either for each observed haplotype within a window or for every permutation of variants observed within a window. This produces a weighted bait set that has higher coverage for more variable regions and reduced bait redundancy for conserved regions. Additionally, pyrad2baits can import individual loci as sequence alignments rather than SNP variant calls. The loci alignments are then passed to aln2baits to generate weighted bait sets,

Candidate baits are then output in FASTA format along with an optional BED file specifying the location of the bait sequences with regards to the input reference sequences.

Quality control and filtration

During the final step, candidate baits are filtered by user-specified quality-control parameters. Filterable parameters include GC content, bait melting temperature, reference sequence base quality, percentage of masked bait sequence, presence of gaps and unknown bases (Ns) in bait sequences, and whether generated baits are shorter than the specified desired bait length. Baits with gap characters can also be extended with flanking sequence to ensure that deletion variants are efficiently captured. BaitsTools then generates a set of filtered baits in FASTA format and an optional BED file describing the location of filtered baits with regard to the input reference sequences. For the vcf2baits and stacks2baits commands, BaitsTools also produces a filtered VCF or Stacks summary file, respectively. For user reference, the filtration parameters are added to the header of the filtered VCF after the BaitsTools variant selection and bait generation parameters. BaitsTools can also generate a summary that tabulates the filtration parameters and inclusion/exclusion from the final filtered bait set for each candidate bait. The quality-controlled baits are suitable either for direct manufacture of hybridization capture kits or further filtration using platform-specific proprietary pipelines.

User interface

To accommodate users with different computational needs and comfort with command-line interfaces, BaitsTools utilizes both a standard command-line interface using arguments and an interactive interface using text prompts (Figure 2). An optional graphical frontend is also available for macOS systems. Executing the baitstools.rb script without subcommands or arguments prints a list of available subcommands and their functions to the screen. Detailed help messages are available for each subcommand by executing the baitstools.rb script with a subcommand and the ‘-h’ or ‘--help’ arguments. Executing the baitstools.rb script with a subcommand without further arguments launches the interactive interface. For instance, executing ‘baitstools.rb vcf2baits –h’ prints detailed help on the vcf2baits subcommand, whereas executing ‘baitstools.rb vcf2baits’ activates the interactive prompts for the vcf2baits subcommand. Furthermore, to improve user-friendliness, BaitsTools will interactively prompt the user to correct entries from the command-line interface when the subcommand cannot be executed as entered (e.g. if a needed input file is not found). Upon execution, BaitsTools will print to the screen the complete interpreted command (including user-unspecified defaults) to ensure that users can accurately reproduce their commands in later analyses.

Software requirements and licensing

BaitsTools is a self-contained Ruby (Matsumoto 2013) package and is therefore compatible with most UNIX and UNIX-like operating systems. Besides Ruby (version 2.0 or greater) and its standard library, BaitsTools has no additional dependencies and does not require local compilation before execution. The optional frontend requires the Ruby gem “tk” (version 1.2 or greater) (Shibata 2017) and the Ruby Version Manager (Seguin & Papis 2016). BaitsTools is compatible with both the Ruby reference implementation (Matz’s Ruby Interpreter) and the Rubinius (version 3.73 or greater) compiler (Phoenix 2006). The program is freely available under the Smithsonian Institution terms of use (http://www.si.edu/termsofuse).

BaitsTools pipelines

Although BaitsTools produces high-quality bait sets on its own, bait set performance can be improved with the addition of external tools into the bait generation pipeline (Figure 3). To reduce the capture of repetitive regions and low complexity sequences, RepeatMasker (Smit *et al*. 2013–2015) can mask these features in the reference sequences. BaitsTools can then exclude baits that include repetitive sequences using the ‘-K’ or ‘--maxmask’ arguments. Downstream of BaitsTools, baits can be clustered using Cd-hit (Li & Godzik 2006) to efficiently remove overly redundant sequences. BLAST (Altschul *et al*. 1990) searches of bait sequences against reference genomes and the other candidate baits can help identify problematic oligonucleotides for removal. Common issues include non-specific baits that can hybridize with multiple genomic targets, self-complementarity, and inter-bait hybridization.

Comparison to existing software

BaitsTools is more flexible and covers a wider variety of hybridization capture applications than existing publicly available software, such as BaitDesigner (Broad Institute 2017) and the PHYLUCE ultra-conserved element (UCE) workflow (Faircloth 2017). BaitDesigner is an unpublished oligonucleotide bait design tool included within the Picard package (Broad Institute 2017). BaitDesigner implements a few features not currently included in BaitsTools (such as Agilent file output). However, BaitDesigner only accepts FASTA sequences as input and has limited bait filtration and quality control options. It also requires the generation of Picard interval lists prior to usage. This interval list can be used to extract regions of interest from the reference sequence. BaitsTools's bed2baits and annot2baits performs similar region extraction without the need for a custom file format.

The PHYLUCE UCE workflow is designed to identify and produce baits for UCE loci from aligned genomes and sequence data (Faircloth 2017). Although BaitsTools does not identify UCEs, it can be used to design appropriate bait sequences once these loci are identified. BaitsTools, however, does not provide the post-capture and sequencing UCE data analysis pipeline included in PHYLUCE (Faircloth *et al*. 2012). Neither BaitDesigner nor the PHYLUCE UCE workflow can design baits from VCFs, LOCI files, or Stacks population summary statistics files.

Performance

To benchmark typical BaitsTools performance, baits were generated and filtered using sequence data from previously sequenced African wild dog (*Lycaon pictus*) genomes (Campana *et al*. 2016), reference sequences from GenBank (accessions: KT448283.1, NC_008093.1, NC_002008.4, NC_006621.3) (Bjornerfeldt *et al*. 2006; Kim *et al*. 1998; Koepfli *et al*. 2015; Lindbad-Toh *et al*. 2005), and simulated Stacks data and ipyrad loci (available: ipyrad.readthedocs.io/output_formats.html). Benchmarked datasets are included in the example data within the BaitsTools repository, except for the *Canis familiaris* X chromosome sequence

(GenBank accession: NC_00621.3) due to file size limitations. All benchmark analyses used BaitsTools version 0.9 and were performed single-threaded on a desktop computer running macOS El Capitan (10.11.6) powered by a 3.5 GHz hexacore Intel Xeon E5 processor with 64 GB 1866 MHz DDR3 ECC memory. Benchmark analyses and results are summarized in Table 2. Benchmark analyses were run under default settings unless otherwise noted. RNA baits were generated to capture DNA sequences. Sequence ambiguities were collapsed. The full-length bait was required. Baits including gaps or unknown bases were excluded. Retained baits' had GC contents between 30% and 50% and melting temperatures were between 0.0°C and 120.0°C. Parameter files, absolute BED coordinates (except in the checkbaits experiment), and detailed logs were output for all experiments.

Furthermore, to compare performance between BaitsTools tilebaits and BaitDesigner (Picard version 2.9.4), baits were generated from a 16,725 bp *Lycaon pictus* mitogenome (GenBank accession: CM007595.1; Campana *et al*. 2016) under analogous settings. Each program generated 120 bp baits with a 60 bp offset between baits. The full-length bait was required. Since BaitDesigner does not filter baits and cannot tile over circular sequences, no other filters were applied in BaitsTools and the mitogenome was treated as a linear sequence. Bait coordinates were output either as an interval list (BaitDesigner) or as a BED file (BaitsTools). BaitsDesigner completed the task in 0.512 wall-clock seconds (0.814 user seconds, 0.092 system seconds), whereas tilebaits finished in 0.120 wall-clock seconds (0.100 user seconds, 0.014 system seconds). The resulting baits were identical.

BaitsTools is fast. Most benchmarking experiments completed in less than a second. Furthermore, tilebaits produced the same bait set as BaitDesigner in 23% of the wall-clock time and 12% of the user time.

Conclusion

BaitsTools is a user-friendly, fast, open-source software package that simplifies the production of baits for hybridization capture. Since the software is highly user configurable and reads a variety of input formats, BaitsTools can produce baits for a wide range of targeted genomics applications.

Acknowledgements

The author thanks the members of the Center for Conservation Genomics, Smithsonian Conservation Biology Institute for their support. The National Science Foundation (award DEB-1547168), the Morris Animal Foundation (award D14ZO-308), and the National Geographic Society (award 8846-10) supported this research.

References

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *Journal of Molecular Biology*, **215**, 403–410.

Bjornerfeldt S, Webster MT, Vilà C (2006) Relaxation of selective constraint on dog mitochondrial DNA following domestication. *Genome Research*, **16**, 990–994.

Broad Institute (2017) Picard 2.11.0. Available from: broadinstitute.github.io/picard/.

Burbano HA, Hodges E, Green RE *et al*. (2010) Targeted investigation of the Neandertal genome by array-based sequence capture. *Science*, **328**, 723–725.

Campana MG, Hawkins MTR, Henson LH *et al*. (2016) Simultaneous identification of host, ectoparasite, and pathogen DNA via in-solution capture. *Molecular Ecology Resources*, **16**, 1224–1239.

Campana MG, Parker LD, Hawkins MTR *et al.* (2016) Genome sequence, population history, and pelage genetics of the endangered African wild dog (*Lycaon pictus*). *BMC Genomics*, **17**, 1013.

Catchen J, Hohenlohe PA, Bassham *et al*. (2013) Stacks: an analysis tool set for population genomics. *Molecular Ecology*, **22**, 3124–3140.

Catchen JM, Amores A, Hohenlohe P *et al*. (2011) *Stacks*: building and genotyping loci *de novo* from short-read sequences. *G3: Genes, Genomes, Genetics*, **1**, 171–182.

Eaton DAR (2014) PyRAD: assembly of *de novo* RADseq loci for phylogenetic analyses. *Bioinformatics*, **30**, 1844–1849.

Faircloth BC (2017) Identifying conserved genomic elements and designing universal bait sets to enrich them. *Methods in Ecology and Evolution*. doi: http://dx.doi.org/10.1111/2041-210X.12754.

Faircloth BC, McCormack JE, NG Crawford *et al*. (2012) Ultraconserved elements anchor thousands of genetic markers for target enrichment spanning multiple evolutionary timescales. *Systematic Biology*, **61**, 717–726.

Gnirke A, Melnikov A, Maguire J *et al*. (2009) Solution hybrid selection with ultra-long oligonucleotides for massively parallel targeted sequencing. *Nature Biotechnology*, **27**, 182–189.

Hawkins MTR, Hofman CA, Callicrate T *et al*. (2016) In-solution hybridization for mammalian mitogenome enrichment: pros, cons and challenges associated with multiplexing degraded DNA. *Molecular Ecology Resources*, **16**, 1173–1188.

Kim KS, Lee SE, Jeong HW, Ha JH (1998) The complete nucleotide sequence of the domestic dog (*Canis familiaris*) mitochondral genome. *Molecular Phylogenetics and Evolution*, **10**, –220.

Koepfli K-P, Pollinger J, Godinho R *et al*. (2015) Genome-wide evidence reveals that African and Eurasian golden jackals are distinct species. *Current Biology*, **16**, 2158–2165.

Li W, Godzik A (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, **10**, 1658–1659,

Lim HC, Braun MJ (2016) High-throughput SNP genotyping of historical and modern samples of five bird species via sequence capture of ultraconserved elements. *Molecular Ecology Resources*, **16**, 1204–1223.

Lindblad-Toh K, Wade CM, Mikkelsen *et al.* (2005) Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, **438**, 803–819.

Ng SB, Turner EH, Robertson PD *et al*. (2009) Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, **461**, 272–276.

Matsumoto Y (2013) Ruby programming language, version 2.0.0. Available from: http://www.ruby-lang.org.

Phoenix E (2006). Rubinius, version 3.73 Available from: https://rubinius.com.

Shibata H (2017). Tk 0.1.0. Available from: https://rubygems.org/gems/tk/versions/0.1.0.

Seguin WE, Papis M (2016). RVM: Ruby Version Manager 1.2.7. Available from: https://rvm.io.

Smit AFA, Hubley R, Green P (2013–2015). RepeatMasker Open-4.0. Available from: http://www.repeatmasker.org.

Data Accessibility

The program, user documentation, tutorial, and example dataset are available on GitHub (https://github.com/campanam/BaitsTools).

Author Contributions

M.G.C. wrote the software and manuscript and performed all analyses.

Tables and figures

Table 1: BaitsTools subcommands, their functions, and input file requirements.

Subcommand	**Function**	**Input formats**
aln2baits	Generate variability-weighted baits from an alignment file	Alignment: FASTA/FASTQ
annot2baits	Generate baits from a genome annotation file and a reference sequence	Annotation: GTF/GFF Reference: FASTA/FASTQ
bed2baits	Generate baits from a BED file and a reference sequence	Features: BED Reference: FASTA/FASTQ
checkbaits	Evaluate and filter previously generated baits	Baits: FASTA/FASTQ
pyrad2baits	Select variants and generate baits from a PyRAD and ipyrad loci files	Loci: LOCI
stacks2baits	Select variants and generate baits from a Stacks population summary statistics file and a reference sequence	Stacks: sumstats.TSV Reference: FASTA/FASTQ
tilebaits	Generate baits from a list of sequences	Sequences: FASTA/FASTQ
vcf2baits	Select variants and generate baits from a VCF file and a reference sequence	Variants: VCF Reference: FASTA/FASTQ

Table 2: BaitsTools benchmarking experiments. The user, system, and wall-clock completion times are listed in seconds. Benchmarked file names are listed at the end of the experiment description in parentheses).

Subcommand	**Experiment description**	**User**	**System**	**Wall-clock**
aln2baits	Weighted baits were generated and filtered from an alignment of five canid mitogenomes (canid_mito_aln.fa).	0.250	0.016	0.276
annot2baits	Baits were generated and filtered for all annotated genes and tRNAs from a *Lycaon pictus* mitogenome (Ananku.fa, Ananku.gff).	0.052	0.011	0.065
bed2baits	Weighted baits were generated and filtered from five 999-bp regions from an alignment of five canid mitogenomes (canid_mito_aln.fa, canid_mito_aln.bed).	0.066	0.012	0.080
checkbaits	Bait quality control was performed on the baits output from the aln2baits benchmarking experiment.	0.0148	0.014	00.0167
pyrad2baits	Baits were generated and filtered from two simulated ipyrad loci treated as sequence alignments (ipyrad.loci).	0.054	0.086	0.017
stacks2baits	Variants were sorted by population and deviation from Hardy-Weinberg Equilibrium ($\alpha = 0.025$; options -H -A 0.025). Up to five variants (option -t 5) per category were selected. No baits were output (option -p) (example.sumstats.tsv).	0.054	0.012	0.070
tilebaits	Baits were generated and filtered from two	0.243	0.014	0.243

	Lycaon pictus mitogenomes and a FASTA file of canid pelage genes (lycaon_mito.fa, pelage_genes.fa).			
vcf2baits	One-hundred *Lycaon pictus* X chromosome sequence variants were selected (option --m 100). Baits were generated and filtered from the selected variants using the *Canis familiaris* reference sequence (NC_00621.3) (WDF20_X.raw.vcf.gz).	988.553	1.248	990.551

Figure 1: BaitsTools workflow. The entry points for each subcommand and the outputs from each BaitsTools step are listed. pyrad2baits is listed twice since it can treat input LOCI files either as variant-call files or sequence alignments.

Figure 2: BaitsTools interactive interface. Executing the baitstools.rb script without further arguments prints the splash screen detailing the available subcommands (top). Executing the script with a subcommand (but omitting other arguments) starts the interactive interface (bottom). Here the user has started the interactive prompts for the vcf2baits subcommand.

Figure 3: An example pipeline to generate highest-quality oligonucleotide bait sets. Reference sequences are masked with RepeatMasker to remove repetitive and low-complexity sequences. Candidate baits are generated from the masked reference sequences and filtered using BaitsTools. Filtered bait sequences are clustered using Cd-hit. Finally bait sets are interrogated using BLAST searches for features such as inter-bait hybridization.

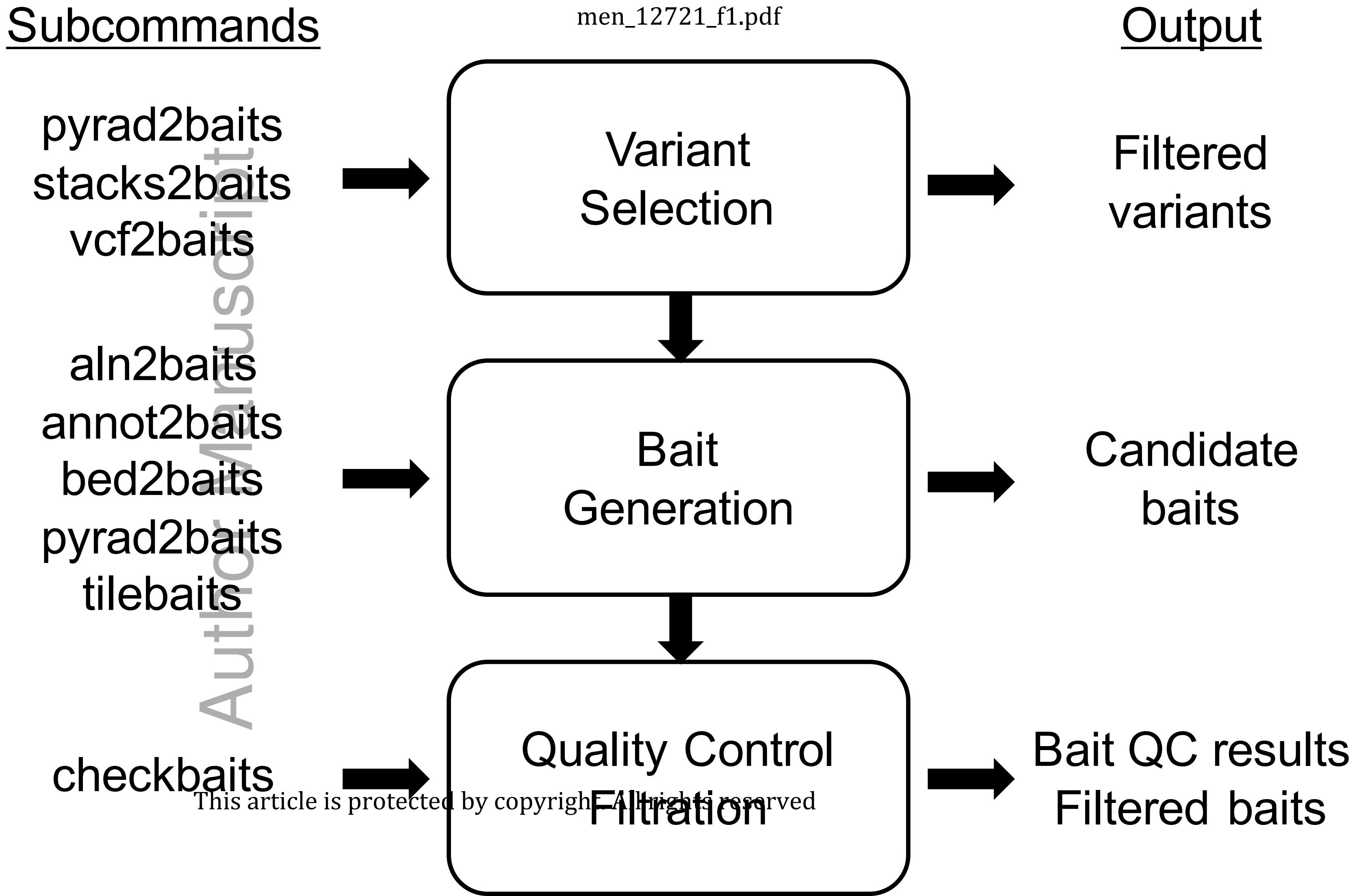
Table 1: BaitsTools subcommands, their functions, and input file requirements.

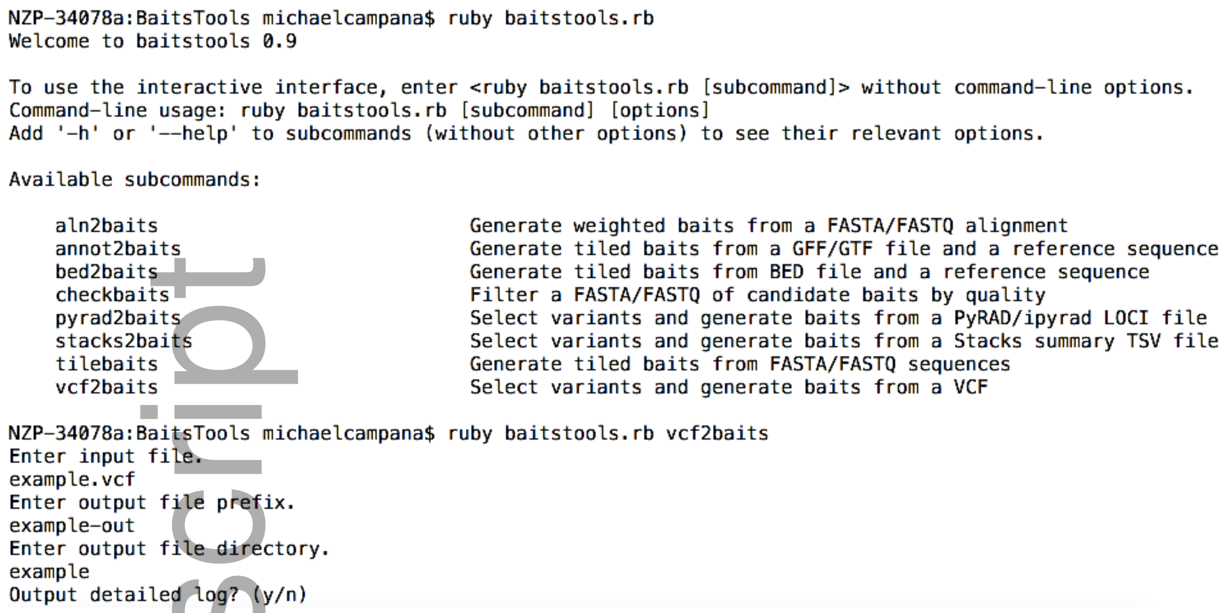
Subcommand	**Function**	**Input formats**
aln2baits	Generate variability-weighted baits from an alignment file	Alignment: FASTA/FASTQ
annot2baits	Generate baits from a genome annotation file and a reference sequence	Annotation: GTF/GFF Reference: FASTA/FASTQ
bed2baits	Generate baits from a BED file and a reference sequence	Features: BED Reference: FASTA/FASTQ
checkbaits	Evaluate and filter previously generated baits	Baits: FASTA/FASTQ
pyrad2baits	Select variants and generate baits from a PyRAD and ipyrad loci files	Loci: LOCI
stacks2baits	Select variants and generate baits from a Stacks population summary statistics file and a reference sequence	Stacks: sumstats.TSV Reference: FASTA/FASTQ
tilebaits	Generate baits from a list of sequences	Sequences: FASTA/FASTQ
vcf2baits	Select variants and generate baits from a VCF file and a reference sequence	Variants: VCF Reference: FASTA/FASTQ

Table 2: BaitsTools benchmarking experiments. The user, system, and wall-clock completion times are listed in seconds. Benchmarked file names are listed at the end of the experiment description in parentheses).

Subcommand	Experiment description	User	System	Wall-clock
aln2baits	Weighted baits were generated and filtered from an alignment of five canid mitogenomes (canid_mito_aln.fa).	0.250	0.016	0.276
annot2baits	Baits were generated and filtered for all annotated genes and tRNAs from a Lycaon pictus mitogenome (Ananku.fa, Ananku.gff).	0.052	0.011	0.065
bed2baits	Weighted baits were generated and filtered from five 999-bp regions from an alignment of five canid mitogenomes (canid_mito_aln.fa, canid_mito_aln.bed).	0.066	0.012	0.080
checkbaits	Bait quality control was performed on the baits output from the aln2baits benchmarking experiment.	0.0148	0.014	00.0167
pyrad2baits	Baits were generated and filtered from two simulated ipyrad loci treated as sequence alignments (ipyrad.loci).	0.054	0.086	0.017
stacks2baits	Variants were sorted by population and deviation from Hardy-Weinberg Equilibrium (α = 0.025; options -H -A 0.025). Up to five variants (option -t 5) per category were selected. No baits were output (option -p) (example.sumstats.tsv).	0.054	0.012	0.070
tilebaits	Baits were generated and filtered from two Lycaon pictus mitogenomes and a FASTA file of canid pelage genes (lycaon_mito.fa, pelage_genes.fa).	0.243	0.014	0.243

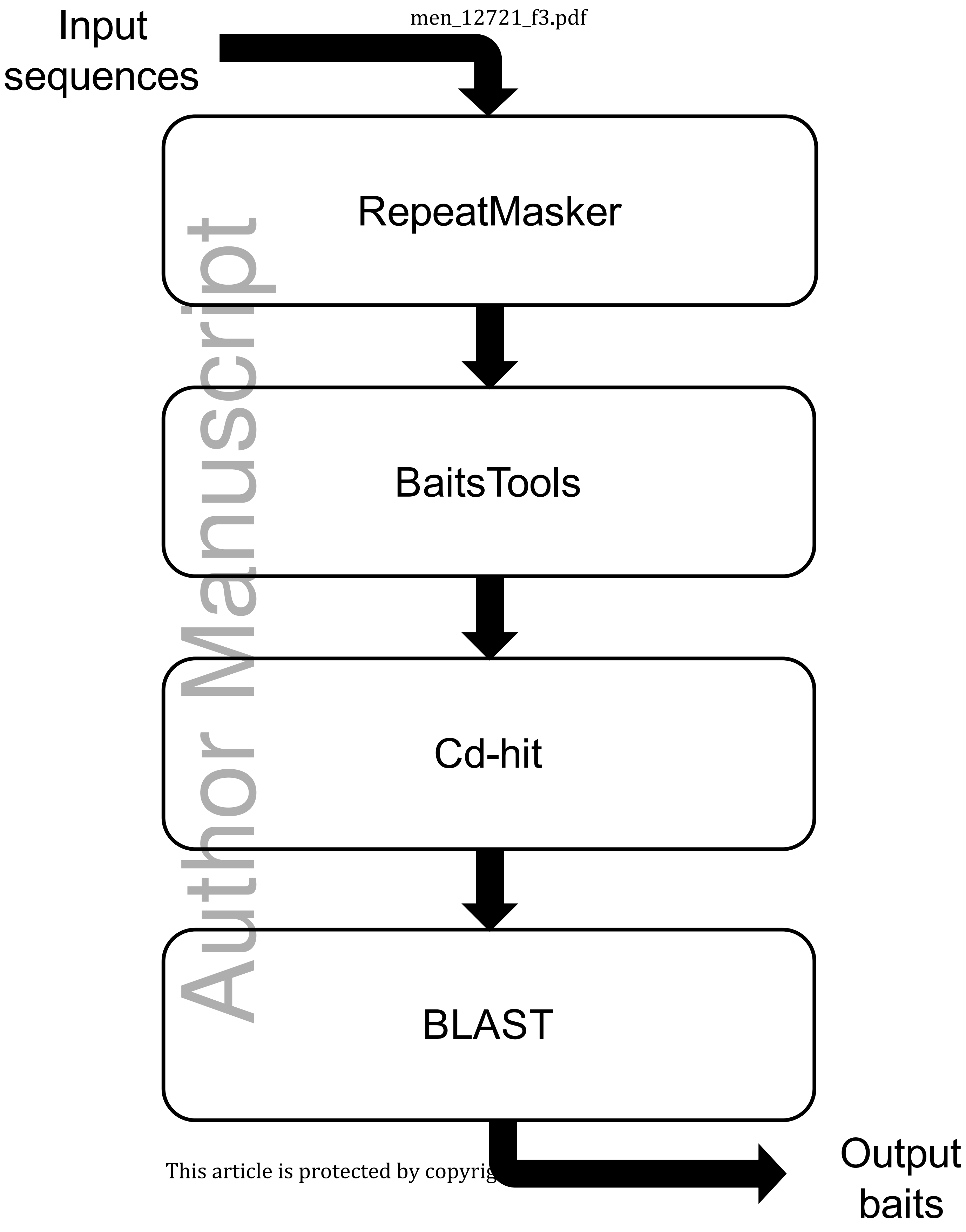
vcf2baits	One-hundred Lycaon pictus X chromosome sequence variants were selected (option --m 100). Baits were generated and filtered from the selected variants using the Canis familiaris reference sequence (NC_00621.3) (WDF20_X.raw.vcf.gz).	988.553	1.248	990.551


Subcommands
Output
pyrad2baits
stacks2baits
vcf2baits
Variant Selection
Filtered variants
aln2baits
annot2baits
bed2baits
pyrad2baits
tilebaits
Bait Generation
Candidate baits
checkbaits
Quality Control Filtration
Bait QC results
Filtered baits



```
NZP-34078a:BaitsTools michaelcampana$ ruby baitstools.rb
Welcome to baitstools 0.9

To use the interactive interface, enter <ruby baitstools.rb [subcommand]> without command-line options.
Command-line usage: ruby baitstools.rb [subcommand] [options]
Add '-h' or '--help' to subcommands (without other options) to see their relevant options.

Available subcommands:

    aln2baits                            Generate weighted baits from a FASTA/FASTQ alignment
    annot2baits                          Generate tiled baits from a GFF/GTF file and a reference sequence
    bed2baits                            Generate tiled baits from BED file and a reference sequence
    checkbaits                           Filter a FASTA/FASTQ of candidate baits by quality
    pyrad2baits                          Select variants and generate baits from a PyRAD/ipyrad LOCI file
    stacks2baits                         Select variants and generate baits from a Stacks summary TSV file
    tilebaits                            Generate tiled baits from FASTA/FASTQ sequences
    vcf2baits                            Select variants and generate baits from a VCF

NZP-34078a:BaitsTools michaelcampana$ ruby baitstools.rb vcf2baits
Enter input file.
example.vcf
Enter output file prefix.
example-out
Enter output file directory.
example
Output detailed log? (y/n)
```

men_12721_f2.tiff


Input sequences
RepeatMasker
BaitsTools
Cd-hit
BLAST
Output baits