mrbait Documentation

Release 1.1.2

Tyler K. Chafin

Jun 17, 2018

Contents

mrbait is a software pipeline for identifying regions of interest in DNA sequence data and designing probes to enrich them.

The motivation behind mrbait is ease and flexibility of use. As such, mrbait allows a variety of input types and facilitates a diverse array of bait design approaches, such as those targeting ultraconserved elements, RAD-capture methods, or those targeting exons or other genomic elements. mrbait also enables fast and efficient iterative design (e.g. to explore parameter settings) using native Python parallelization and an SQL database back-end. In this documentation, you can learn about the overall process employed by mrbait (Pipeline), how to install mrbait for use on a personal desktop or remote workstation or HPC (Getting Started), see a full description of all runtime options (Running mrbait), and see walltime and memory benchmarking results (Benchmarking)

mrbait code is open-source and freely available at on GitHub

Official releases can be found here

Having issues running or installing mrbait? Contact me at tkchafin@uark.edu or post an Issue on the GitHub page.

Citation: A manuscript describing mrbait and it's underlying methodology is currently in review. For now, please cite the GitHub repository.

CHAPTER 1

Introduction

A variety of genome reduction methods have been implemented to reduce costs of applying next-generation sequencing methods to non-model organisms, or projects with large numbers of samples (e.g. those focusing on the population scale). These can broadly be classified into those which use restriction enzymes and size selection for subsampling genomic complexity [e.g. RADseq methods (Baird et al., 2008; Peterson et al., 2012)], and those which enrich for fragments selected a priori using biotinylated RNA 'baits' (Lemmon et al., 2012; McCormack et al., 2012). The latter benefit from increased specificity, yet require some genomic information for marker development. To mitigate, some take a hybrid approach by using baits to enrich RAD loci which are most consistently recovered, or to maximize capture of parsimony-informative variation (Ali et al., 2016; Hoffberg et al., 2016).

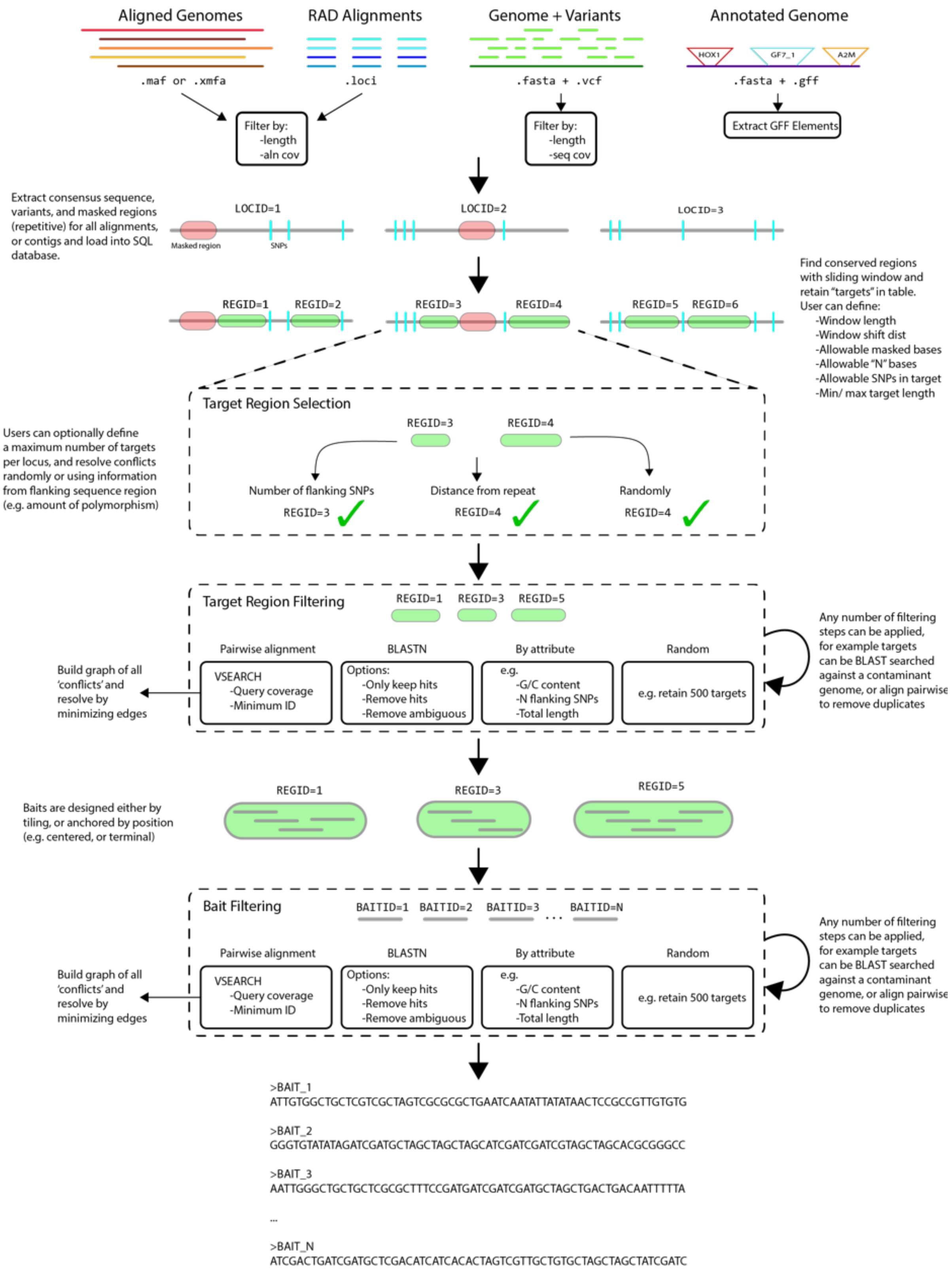
Applying these targeted-enrichment methods (either via RAD-capture methods, ultra-conserved elements, or anchored-enrichment) requires first bioinformatic processing to parse large alignments, identify candidate regions for bait-design, and design of complementary oligonucleotide sequences for synthesis. Although some developers are transparent in providing computational resources and workflows to design such probe sets (e.g. see Faircloth 2017), a generalized and flexible pipeline does not yet exist. The motivation behind MrBait was to provide such a resource, which could be universally applied to differing bait enrichment strategies (e.g. targeting ultra-conserved regions vs. functional elements), and facilitate diverse quality control methods to mitigate non-target capture (contamination, etc), target-target hybridization, ambiguous mapping, and enrichment of repetitive DNA.

CHAPTER 2

Pipeline Description

The general process (summarized in figure below) is built on a relational database in SQLite, populated, accessed, and parsed in Python. It takes a variety of input file formats, and is written modularly such that adding additional capabilities (e.g. input file formats, filtering schema) can be done without too much difficulty. The workflow is divided into 5 steps, as follows:

1. Alignments (provided as .xmfa, .loci, or .maf) or genomes (provided as .fasta, annotated with .vcf or .gff) will be used to build a consensus sequence of each locus.
2. A sliding window will be applied to each consensus to find candidate targets for which baits could be designed
3. Targets are then selected (if too close together, or only one allowed per locus), and filtered according to any number of specified filter (e.g. GC content, flanking SNPs, pairwise alignment)
4. Passing targets are then parsed to design a putative set of baits
5. Baits are then filtered according to selected criteria, and output as FASTA.



Extract consensus sequence, variants, and masked regions (repetitive) for all alignments, or contigs and load into SQL database.

Find conserved regions with sliding window and retain "targets" in table.
User can define:
-Window length
-Window shift dist
-Allowable masked bases
-Allowable "N" bases
-Allowable SNPs in target
-Min/ max target length

Users can optionally define a maximum number of targets per locus, and resolve conflicts randomly or using information from flanking sequence region (e.g. amount of polymorphism)

Build graph of all 'conflicts' and resolve by minimizing edges

Any number of filtering steps can be applied, for example targets can be BLAST searched against a contaminant genome, or align pairwise to remove duplicates

Baits are designed either by tiling, or anchored by position (e.g. centered, or terminal)

Build graph of all 'conflicts' and resolve by minimizing edges

Any number of filtering steps can be applied, for example targets can be BLAST searched against a contaminant genome, or align pairwise to remove duplicates

```
>BAIT_1
ATTGTGGCTGCTCGTCGCTAGTCGCGCGCTGAATCAATATTATATAACTCCGCCGTTGTGTG

>BAIT_2
GGGTGTATATAGATCGATGCTAGCTAGCTAGCATCGATCGATCGTAGCTAGCACGCGGGCC

>BAIT_3
AATTGGGCTGCTGCTCGCGCTTTCCGATGATCGATCGATGCTAGCTGACTGACAATTTTTA

...

>BAIT_N
ATCGACTGATCGATGCTCGACATCATCACACTAGTCGTTGCTGTGCTAGCTAGCTATCGATC
```

CHAPTER 3

Getting Started

mrbait has been tested on Mac and Linux operating systems and is primarily supported on those platforms. However, Windows users can easily install using the built-in Linux subsystem for Windows 10.

In-development code can be found on the Github page: https://github.com/tkchafin/mrbait

If you find any issues with the program, please email me at tkchafin@uark.edu or submit as an issue on Github, which can also be used for submitting feature requests. When submitting bugs or issues, please include input files, your command-line call, and any output MrBait produced to the screen or output files.

3.1 Availability

Functioning releases can be found at: https://github.com/tkchafin/mrbait/releases

3.2 Dependencies

mrbait is written for Python3, and requires Python version >= 3.6.0. The recommended method of acquiring Python and all other dependencies is via the Anaconda distribution, as outlined in Section 3.3. A full list of dependencies is given below.

Python >= 3.6 SQLite3 BioPython Pandas >=0.22 numpy pyVCF networkx

mrbait can optionally use the following programs during bait development:

blast vsearch

For these utilities, please cite the following: Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL. 2009. BLAST+: architecture and applications. BMC Bioinformatics. 1-:410. Doi:10.1186/1471-2105-10.421

Rognes R, Flouri T, Nichols B, Quince C, Mahe F. 2016. VSEARCH: A versatile and open source tool for metagenomics. PeerJ. 4:e2584. Doi: 10.7717/peerj.2584

3.3 Installation

By far the easiest way to acquire and install mrbait is via conda, a command line interface for managing and installing packages. Download and install anaconda for Python 3.6 here: https://www.anaconda.com/download/. If you are wanting a minimal environment, or a faster install, you can also use the Miniconda distribution (https://conda.io/miniconda.html) with the same commands. After installation, be sure to test that conda is installed by typing **conda info**, which will print information about your installation. Note, you may first need to reload your bash environment by typing **source ~/.bashrc** or **source ~/.bash_profile** on Mac. Assuming success, the installation process is then very straightforward:

```
#This command tells conda that the code and dependencies for mrbait can
#be found in 'channels' bioconda, conda-forge, and tylerkchafin.
conda install mrbait -c tylerkchafin -c bioconda -c conda-forge

#If you would like to instead install the latest development version, you can
#clone the github repository and
#install MrBait like so (assuming you have git installed):
git clone https://github.com/tkchafin/mrbait.git
cd mrbait
python ./setup.py install
```

You will then need to manually install both vsearch and blast, only if you install directly from the GitHub source using the setup.py installation. These will be installed for you if you used conda.

Windows users: MrBait is installable using the built-in Linux subsystem for Windows 10. I have only tested using the Ubuntu OS subsystem configuration but assume that other Linux distros would work equally well. If you prefer, you can also use a Linux installation on a virtual machine, or installed portably on a USB-attached drive, although this may impact performance. Contact me at tkchafin@uark.edu if you have any issues getting mrbait installed, or feel free to launch an 'Issue' on the GitHub page.

HPC users: One of the reasons I recommend using conda to manage your Python environment, is that it keeps your packages separate from the system environment, which you often will not have permissions to modify. Anaconda will instead install your own local flavor of Python in your home directory, where is will also install any additional packages you choose to add.

BLAST and VSEARCH: conda will also install both BLAST and VSEARCH and place them within your conda environment. If you would like to manually manage versions of these programs, or use an existing installation, you can provide the paths to those binaries using the *–vsearch* and *–blastn* commands for mrbait.

3.4 Running mrbait

Assuming you have completed the recommended conda install, mrbait and it's *Dependencies* should already be in your path and is now fully ready to go. You can verify successful installation, and view the help menu, by typing: mrbait -h

Instructions for bait design are provided as arguments (see Section 5 for thorough usage instructions, and Section 8 for tutorials). For example, to generate baits of length 80, tiled across target regions with an overlap of 40 bases, from a Multiple Alignment File (MAF) "example.maf":

```
mrbait -M example.maf -b 80 -s tile=40
```

Or, to also filter for only alignments including 5 or more individuals, and of length >500:

```
mrbait -M example.maf -b 80 -s tile=40 -l 500 -c 5
```

CHAPTER 4

Input files

This section describes the input file types accepted by MrBait.

4.1 Assembled genomes

mrbait only accepts genome assemblies formatted as FASTA. These can represent contigs, scaffolds, or entire chromosomes. According to the FASTA specifications, a sequence should begin with a header line, or short description (indicated by the ">" symbol), followed by a second line containing sequence data. It does not matter if the following lines are interleaved or on a single line, and any blank lines in the file will be ignored, as will any leading or trailing whitespace.

An example FASTA-formatted sequence is given below.

```
>chr1.scaffold1
ATAGCTCGGCTACGTGATCGCGTGCTC-ATGCTAGCGCTNNNNNNNNATGATTGCTTTT
TGTGTGTGCAAGCACTGCCGRGCTACGCGCTACTGCCRCCTAGTATGTGTGGCCGCTAC
TAGTCCGCGCTAGCTtTtagatctcgtggcgccgcgcgcgtcgcacgatcgtacgcgcc
>chr1.scaffold2
ATCGTGCTGCGGCGCTGCCTCAGC...
...
...
...
```

4.1.1 Annotating genomes with VCF

mrbait also supports supplementing genomic sequences with coordinate-reference SNP data (e.g. obtained from population-level sequencing) using the Variant Call Format:

```
##fileformat=VCFv4.2
##FORMAT=<ID=GT,Number=1,Type=Integer,Description="Genotype">
##FORMAT=<ID=GP,Number=G,Type=Float,Description="Genotype Probabilities">
```

(continues on next page)

(continued from previous page)

```
##FORMAT=<ID=PL,Number=G,Type=Float,Description="Phred-scaled Genotype Likelihoods">
#CHROM       POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  SAMP001␣
↪SAMP002
chr1.scaffold1       48      rs11449 G       A       .       PASS    .       GT      ␣
↪0/0     0/1
chr1.scaffold1       47      rs11449 T       A       .       PASS    .       GT      ␣
↪0/0     0/1
chr1.scaffold2       1 rs84825 A     T       .       PASS    .       GT:GP   0/1:.   ␣
↪0/1:0.03,0.97,0
...
...
```

It is important to note that the VCF format can communicate much more information than mrbait will utilize. The CHROM and POS columns will be parsed to locate the reference position for each SNP, and the REF and ALT columns to write a new consensus base at that position using IUPAC ambiguity codes (e.g. C/T = Y). More functionality will be added in future versions of mrbait.

It is highly recommended you add variant data if it is available, as it will be used both for finding adequately conserved regions for bait design, as well as for filtering target regions for those which capture flanking SNPs.

4.1.2 Annotating genomes with GFF

mrbait can also make use of genomic features provided using the Generic Feature Format (GFF), independently or in addition to any variant data provided via VCF. mrbait assumes that input GFF files follow the version 3 *GFF specification <https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md>*_:

```
##gff-version 3
chr1.scaffold1       .       gene    10      180     .       +       .       ␣
↪ID=gene0001;Alias=targets
chr1.scaffold1       .       mRNA    20      180     .       +       .       ␣
↪ID=mrna0001;Parent=gene0001
chr1.scaffold1       .       exon    10      128     .       +       .       ␣
↪ID=tfbs00001;Parent=gene0001
...
...
```

Columns should be separated by tabs and defined according to the GFF3 standard (e.g. column 1 contains the sequence ID). mrbait will use the sequence ID (column 1) to map coordinates in GFF columns 4 and 5 to the reference provided in your FASTA file, thus these identifiers must be identical. mrbait will also categorize features internally by the type (e.g. "exon") given in column 3, and by any alias assigned in the attributes column (column 9). All other columns are ignored. You can use either type or alias to tell mrbait to target those features for bait design.

If you are not targeting all of a single type (e.g. CDS, or exon), you can either pre-filter your GFF file prior to loading, or you can annotate features of interest using the Alias attribute.

4.2 Multiple genome alignments

mrbait reads two different input file types for multiple genome alignments. These can be provided using the Multiple Alignment Format (MAF), or the eXtended Multi-FastA (XMFA) formats.

The MAF format is output by several multiple alignment programs, including MAFFT and Mugsy, and take the following general form:

```
##maf version=1 scoring=tba.v8
# tba.v8 (((human chimp) baboon) (mouse rat))
# multiz.v7
# maf_project.v5 _tba_right.maf3 mouse _tba_C
# single_cov2.v4 single_cov2 /dev/stdin

a score=5062.0
s hg16.chr7    27699739 6 + 158545518 RAAAGAGATGCTAAGCCAATGAGTTGATGTCTCTCAATGTGTG
s panTro1.chr6 28862317 6 + 161576975 RAAAGAGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTGTG
s baboon         241163 6 +   4622798 TAAAGAGATGCTAAGCCAATGAGTTGTTGTCTCTRAATGTGTG
s mm4.chr6     53303881 6 + 151104725 TAAAGAGATGCTAAGCCAATGAGTTGTTGTCGCTCAATGTGTG
s rn3.chr4     81444246 6 + 187371129 taaggaGATGCTAAGCCAATGAGTTGTTGTCGCTCAATGTGTG

...
...
...
```

Comment lines (starting with "#") are ignored by mrbait. Alignment blocks (considered by mrbait to each represent different loci) are started with "a", followed by sequence lines starting with "s". Source, strand, and coordinate positions are not informative for mrbait, nor are lines starting with other letters (which can be used in the MAF format to communicate additional information about the preceding sequence, such as quality scores).

The eXtended Multi-FastA (XMFA) format output by the multiple-genome aligner MAUVE (which outputs it as ".alignment") is an extension of the standard FASTA format to allow alignment blocks from many different loci, with header lines representing identifiers for the aligned sequence, and start-end coordinates representing the alignment block location within the genome, followed by the sequence:

```
>1:1-230 +
ATAGC-NAATC--GC...
>2:210-440 -
ATTGGCCAATCCCC...
>3:3-230 +
TTA-CCAAGC--GC...
=
...
...
```

Alignment blocks are delimited by the "=" symbol. All alignment blocks are assumed by mrbait to represent separate, discontinuous loci. Note that because no individual 'alignment block' in the .xmfa file is guaranteed to contain the same genome representatives, no reference coordinates are saved by mrbait. This means that additional annotation via GFF or VCF cannot be added to whole-genome alignments provided in .xmfa format.

4.3 Reduced representation data

Alignments from reduced-representation methods such as restriction-site associate DNA sequencing methods (RAD-seq) can be input using the MAF or XMFA formats, or using the ".loci" format output by the RADseq assembly pipeline pyrad or its successor ipyrad. This format shows individual loci delimited by a line starting with "//" which features additional annotation of variants and parsimony-informative sites:

```
>PopA001     GTGTGATAGTAGTGATGTATTTTATAATATATATTATCGGATAT......
>PopA002     GTGTGARAGTAGTGATGTATTTTATAATATATATTATCGGATAT......
>PopB001     GTGTGACAGTAGTGATGTATTTTATAATATATATTATCGGATAT......
>PopB002     GAGTGATAGTAGTGATGTATTTTATAATATATATTATCGGATAT......
//            *      *                                        |1
```

(continues on next page)

(continued from previous page)

```
...
...
...
```

mrbait ignores annotation information (since it parses variants anyways to generate a consensus sequence), and only uses the "//" delimiter to distinguish between alignment blocks. Creating a .loci file from other formats can be accomplished relatively easily. For example, a series of separate alignments (each as .fasta), could be converted to the **.loci** format using the following **bash** command:

```
for file in `ls example*.fasta`; do
  awk 'BEGIN{ORS=""}$1~/^\>/{print $01"\t";next}{print $0"\n"}' $file
  >> example.loci;
  echo "//" >> example.loci;
done
```

CHAPTER 5

Usage options

mrbait reads all options and inputs using command-line arguments provided after the program name. For a quick look at all options from the command line, call the help menu by typing **mrbait -h** from the terminal.

Note that options requiring a floating point number (e.g. *-q*) allow inputs from 0.0 to 1.0, and options requiring an integer (e.g. *-c*) allow inputs ranging from 1 to infinity.

5.1 Main Parameters

5.1.1 General options

-r, --resume

Resume: This flag is used to tell mrbait if you would like to resume work following a particular step. Use this option in conjunction with the **–db** flag to continue the pipeline if you would like to re-perform filtering steps without needing to re-load and parse alignments

Usage: -r 1: Continue pipeline after Step 1 (loading alignments) -r 2: Continue fter Step 2 (target discovery) -r 3: Continue after Step 3 (target filtering) -r 4: Continue after Step 4 (bait discovery) For example, -r 4 will tell mrbait to re-do bait filtering and output

--db

Database: Use this with the –resume flag to specify a .sqlite database file from which to start the pipeline.

-T, --threads

Threads: Number of threads to use with processes that run in parallel. This will also be passed to vsearch and/or blast if those are being called. [default=1]

-h, --help

Help: Exit and display the help menu

5.1.2 Input Options

-M, --maf

MAF input: Use this to provide the path to the multiple alignment MAF file

-X, --xmfa **XMFA input**: As an alternative to the MAF file, you can provide the .xmfa file output by the aligner Mauve.

-L, --loci **LOCI input**: Multiple alignments can also be provided using the .loci file output by the RADseq assembly pipeline pyRAD.

-A, --assembly **FASTA input**: Genome assembly provided as FASTA

-V, --vcf **VCF input**: For use with –assembly: VCF file containing variant data

-G, --gff **GFF input**: For use with –assembly: GFF file containing feature data

5.1.3 Alignment filtering/ consensus options (use with -M, -X, -L)

-c, --cov **Coverage**: Minimum number of individuals/sequences per alignment, for MAF, XMFA, or LOCI inputs [default=1]

-l, --len **Minimum length**: Minimum alignment length to attempt bait design [default=80]

-q, --tresh **Bad base threshold**: Threshold proportion of gaps or N (ambiguous or poor quality) characters to over-ride the consensus base. For example, *-q 0.2* would be interpreted as 20% of bases at a nucleotide position must be an “N” or gap character in order for that character to be represented as the consensus base. [default=0.1]

-Q, --max_ambig **Max bad bases**: Maximum allowable proportion of gap/N characters allowed in a consensus sequence before it will be discarded. *-Q 0.5* means a consensus sequence can be 50% N’s or gap characters (“-“) before being dropped from consideration. [default=0.5]

-k, --mask **Mask threshold**: Threshold proportion of masked characters per nucleotide column to mask the consensus base call. For use when case represents masking information (where lowercase = masked), as when using the *-xsmall* option in RepeatMasker to flag low-complexity or repetitive sequences. Case will be retained in the consensus on a per-base basis according to this threshold. [default=0.1]

-K, --max_mask **Max masked bases**: Maximum allowable proportion of masked characters allowed in a consensus sequence before it will be discarded. *-K 0.5* means a consensus sequence can be 50% masked (lowercase) before being dropped from consideration.[default=0.5].

If lowercase bases do not contain masking information, set to *-K 1.0*

5.1.4 General Bait Design Options

-b, --bait **Bait length**: This is the length of desired baits, and will be used for bait design as well as the sliding window width for target region discovery [default=80]

-w, --win_shift **Sliding window shift distance**: Shift distance for sliding window used to discover target regions. Generally, there should not be a reason to alter this. If target discovery (step 2) is taking a very long time, adjusting this may make it faster although it could result in more targets failing filtering [default=1]

-v, --var_max **Maximum SNPs per bait**: Maximum allowable variants allowed in a bait sequence. These can be expanded in the final output as each possible non-ambiguous bait sequence for synthesis. Use this when there are not enough conserved regions to capture enough loci for your design. [default=0]

-n, --numN **Maximum Ns per bait**: Maximum allowable ambiguous (N) bases allowed per bait. This could be increased when there are too many poor quality bases in your alignment to design a sufficient number of probes, although keep in mind this will affect the specificity of your resulting probes. [default=0]

-g, --numG **Maximum gaps per bait**: Maximum allowable gap characters allowed per bait. If dealing with alignments containing many indels, it might be desirable to allow a small number per bait sequence. These can be expanded in the final output using the -x,–expand option, which will expand gap characters as A, G, T, C, and absent. [default=0]

5.1.5 Target Region Options

These are options primarily used to control which **target regions**, or regions which could be enriched (e.g. conserved enough to design baits) will be used to design the final set of bait sequences. Targets will be either *passed* or *failed* depending on these criteria.

For example, you can constrain targets to fail if they are below 200 bases in length, or above 5000 based in length by specifying *-F len=200,5000*. In this case, all targets with total lengths outside of this range will fail, and be excluded from bait design.

-R, --mult_reg **Multiple targets per locus**: By default, mrbait only chooses one target region (e.g. conserved region for which baits could be designed) per locus/ alignment. When multiple are discovered, they are ranked according to the criterion selected with the *-S,–select_r* option. When *-R,–mult_reg* not in use, only a single target region (and corresponding baits) is chosen per alignment. [default=false]

-m, --min_mult **Minimum length for multiple targets**: Specify this to set a minimum alignment or locus length to allow multiple target regions to be selected. By default will be set to the value of *-l,–len* (thus, when *-R,–mult_reg* is used, all loci passing length filter will be allowed multiple targets).

-D, --dist_r **Distance between targets**: When *-R,–mult_reg* is in use, use this parameter to specify the minimum distance between targets. When targets are in conflict (e.g. they are less than *-D,–dist_r* bases apart), conflicts will be resolved using the criterion set with *-S,–select_r*. [default=100]

-d, --flank_dist **Flanking distance for target filtering**: Distance from boundaries of target region to parse for counting SNPs, ambiguities, gaps, etc when filtering target regions (see *-S,–select_r* and *-F,–filter_r*) [default=500]. Note that this value will tell mrbait to search '*d*' bases to the left AND right of each target region.

Note that currently, the same *–flank_dist* value will be used for all filters.

-S, --select_r **Target selection criterion**: Method to resolve conflicts when targets are too close together (e.g. when *-R* and *-D*), or when only choosing one target per locus/alignment.

Usage: *-S snp*: Select target with most SNPs within *d* bases *-S bad*: Select target with least gaps/Ns within *d* bases *-S cons*: Select target with least SNPs within *d* bases *-S rand*: Randomly select a target [default]

Example: *-d 100 -S snp* to choose region with most SNPs within 100 flanking bases

-F, --filter_r **Target filtering criteria**: Method(s) used to filter all target regions. Can be specified any number of times to use additional filtering criteria.

Usage: *-F len=[x,y]*: Length between *x* (min) and *y* (max) *-F gap=[x]*: Maximum of *x* indels in target region *-F bad=[x]*: Maximum of *x* N characters in target region *-F snp=[x,y]*: Between *x* (min) and *y* (max) SNPs w/in *d* *-F mask=[x]*: Maximum of *x* N characters in target region *-F gc=[x,y]*: G/C propotion between *x* (min) and *y* (max) *-F rand=[x]*: Randomly retain *x* targets *-F pw=[i,q]*: Pairwise alignment, removing when *i* percent identity over at least *q* proportion of the sequences *-F blast_i=[i,q]*: Only retain BLAST hits with *i* percent identity over at least *q* query coverage *-F blast_i=[i,q]*: Exclude BLAST hits with *i* percent identity over at least *q* query coverage *-F gff=[type]*: Only retain targets within *d* bases of a GFF-annotated feature of type type. Only for use when *-A* and *-G* inputs provided. Use *-F gff=all* to target any type of annotated feature. *-F gff_a=[alias]*: Only retain targets within *d* bases of a GFF-annotated feature of tagged with the Alias attribute matching alias. Only for use when *-A* and *-G* inputs provided.

Examples: *-F snp=1,10 -d 100* to sample when 1-10 SNPs within 100 bases *-F gc=0.2,0.8 -F rand=100* to keep 100 random targets w/ 20-80% GC *-F mask=0.1* to remove targets with >10% masked bases *-d 1000 -F gff=exon* to keep targets within 100 bases of an exon

5.1.6 Bait Selection Options

These options are used to specify how baits will be designed in passing target regions (*-s, –select_b*) and how designed baits will be curated to create the final set of sequences for synthesis (*-f, –filter_b*).

The default behavior is to tile baits across all *passing* target regions, with an overlap of 50%. This corresponds to ~2X coverage across the target region. Only passing baits will be included in the final output FASTA file.

-s, --select_b

Bait selection scheme: Use this to specify the desired method to design baits from passing target regions.

Usage: *-s tile=[x]*: Tile baits over whole region, with *x* overlap *-s center=[n,x]*: *n* centered baits with *x* overlap *-s flank=[n,x]*: *n* terminal baits (each end) with *x* overlap Default behavior is to tile baits across all targets with 50% overlap

-f, --filter_b

Bait filtering criteria: Method(s) used to filter baits. Can be specified any number of times to use additional filtering criteria.

Usage: *-F mask=[x]*: Maximum of *x* N characters in target region *-F gc=[x,y]*: G/C propotion between *x* (min) and *y* (max) *-F rand=[x]*: Randomly retain *x* targets *-F pw=[i,q]*: Pairwise alignment, removing when *i* percent identity over at least *q* proportion of the sequences *-F blast_i=[i,q]*: Only retain BLAST hits with *i* percent identity over at least *q* query coverage *-F blast_i=[i,q]*: Exclude BLAST hits with *i* percent identity over at least *q* query coverage

5.1.7 Output Options

Use these options to control the format of your output file, or to specify non-default output files.

-x, --expand

Bait format: Boolean. Use this flag if you want any ambiguities in bait sequences to be expanded (e.g. N = A,G,C,T). IUPAC codes will be fully expanded, and gap characters will be expanded as all nucleotides and as absent. Bait sequences will be output as FASTA to $out_baits.fasta

-t, --print_tr

Print target regions: Boolean. Use this flag if you would like target regions to be printed to a FASTA file. FASTA headers will reflect locus number, target number within locus, and pass=T or pass=F indicating if the target passed or failed filtering specified using the *-F,–filter_r* options, as|br| well as any targets which were excluded due to -S,–select_r criteria. Output file will be named as $out_targets.fasta.

--strand **Output strand**: Use this if you want to print baits as-designed, or as reverse complement. Possible values: “+” [default], “-” (reverse-complement), or “both”.

-o, --out **Output prefix**: Desired prefix for output files. Default is “out”.

5.2 Filtering using vsearch

Using the –filter_r or –filter_b ‘pw’ (for pairwise-align) options call an external open-source package vsearch. Internally, this is accomplished using the “*allpairs_global*” function in vsearch, which performs all-vs-all global alignments of target or bait sequences. mrbait then parses the output of vsearch to find pairs of sequences with greater than *i* percent identity over at least *q* percent of the query sequence. Sequences are first sorted by length, meaning the *q* proportion should be measured by the shorter of the two sequences in each pairwise alignment (although when called with the *–filter_b* command, all pairs are of equal length). Because all sequences will be compared in a pairwise fashion, keep in mind that this step could take considerable time, although VSEARCH is exceptionally efficient and does take advantage of multiple cores for parallel computation (passed using the –threads argument).

If you did NOT install vsearch using the conda installation instructions for mrbait, you may need to point mrbait to the vsearch executable. You can specify this path, as well asspecify how mrbait parses the results of vsearch using the following parameters:

5.2.1 vsearch Options

--vsearch **VSEARCH binary**: Path to vsearch binary. If installed via conda install, the bioconda recipe will be used, and placed into the bin/ folder for your conda installation. mrbait, by default, assumes that the vsearch executable is locatable in your $PATH as “vsearch”. If this is not the case, provide an alternate path using this flag.

--vthreads **VSEARCH threads**: Number of threads to use for vsearch. By default, this is assumed to be the same value as the *–threads* argument passed to mrbait

--noGraph **No conflict graph**: By default, mrbait will try to recover as many sequences as possible from the pairwise alignment results by using a graph structure (see below) to remove only enough sequences to resolve all conflicts

--noWeightGraph **Unweighted conflict resolution**: By default, mrbait weights all nodes by the number of SNPs that the corresponding target or bait captures. Use this option to turn this off, and instead use the built-in maximal_independent_set function from the networkx package.

--weightByMin **Weight by minimum ambiguity**: Instead of trying to keep sequences with the most flanking SNPs, this option tries to keep sequences with the fewest gap or N characters in flanking sequences.

--weightMax **Maximum size to attempt weighting**: Maximum graph size to attempt weighted algorithm for finding maximal independent set. Graphs larger than this size will use the maximal_independent_set function from the networkx package, which is slightly faster.

5.2.2 Graph-based conflict resolution

After vsearch is complete, mrbait will parse the output and represent all 'conflicting' sequences (e.g. those which aligned with i or greater identity and q or greater query coverage) as a graph structure, with nodes as the primary key (ID) for the SQLite entry corresponding to each sequence, and edges unweighted and representing 'conflicts' from the pairwise alignment. By default, mrbait uses this graph structure to attempt to rescue as many sequences as possible which failed the pairwise alignment filter by using an algorithm to find the most sequences which can be "kept" while removing all edges (conflicts) from the graph (e.g. see Figure 2 below).

This is accomplished by searching through all edges, removing the node (target or bait) which captures the least flanking SNPs in the original alignments, or if this value is equal by removing the node with the most neighbors (=conflicting sequences). If both quantities are equal, one is chosen at random. Options are provided to turn off this behavior altogether, or to weight nodes according to how many gap/N characters flank them, or to alternatively use the maximal_independent_set function from the networkx package. Unfortunately, finding the maximal independent set of a graph (i.e. the maximum number of nodes to retain while removing all edges) is an extremely hard problem, and cannot be done in a particularly efficient manner. Thus, with very large sets of target sequences, this can take quite a while if there is a lot of highly matching alignments among them.

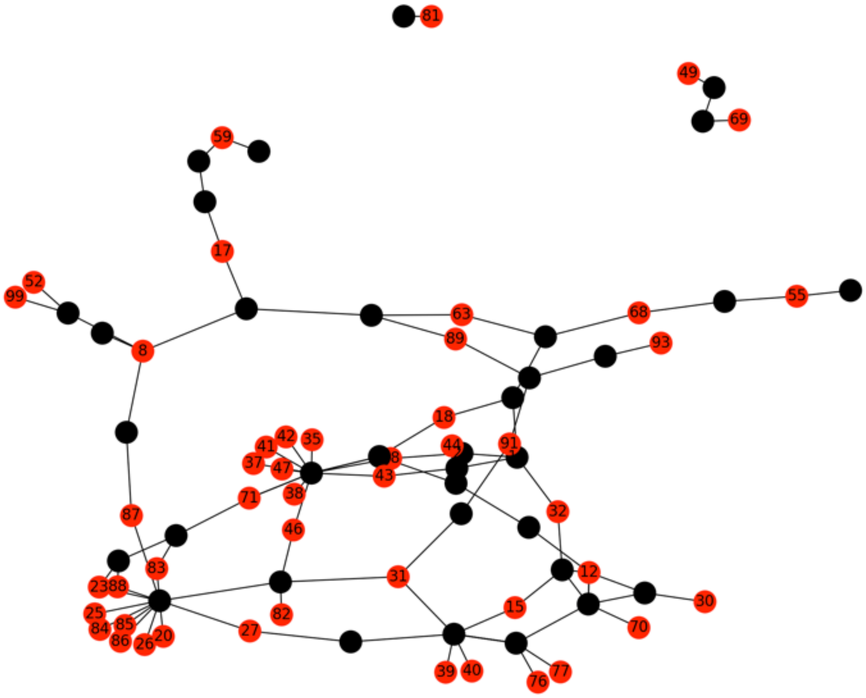


Figure 2: Example maximal independent set (plotted using the matplotlib for Python!) where nodes in red are those which have been kept by the algorithm. Black nodes represent sequences which would be deleted (e.g. baits which fail the 'pw' filter). Although red nodes may be indirectly connected by black 'failed' nodes (i.e. by multiple edges), no direct edges remain in the graph after removing failed nodes.

5.3 Filtering using blast

The NCBI-BLAST+ package (https://blast.ncbi.nlm.nih.gov/Blast.cgi) can also be called using a similar scheme of setting a threshold of *i* sequence identity of hits, and *q* query coverage. Currently, mrbait supports blastn filtering using these cutoff values to either exclude (*blast_x*) or include (*blast_i*) sequences based on presence of hits.

Usage of the '*blast_x*' option for either target (*–filter_r*) or bait (*–filter_b*) filtering would most generally be used to prevent non-target matches to potential contaminant DNA, given a representative genome for the putative non-target organism (for example a common bacterial contaminant). If such a reference genome is lacking, you can also increase specificity of chosen targets or baits using the '*blast_i*' option, which only retains queries with hits to a specified genome.

When using either '*blast_x*' or '*blast_i*' options for filtering, a database file must be specified either as a pre-built BLAST-formatted database using the *–blastdb* flag, or alternatively as a FASTA file using *–fastadb*. In the latter case, mrbait will call the NCBI MakeBlastDB executable. When using the conda install, this is included with all other BLAST+ binaries (including the blastn executable also required by mrbait for BLAST filtering). If not using the conda installation instructions, these are both assumed to be accessible in your $PATH as '*blastn*' and '*makeblastdb*'.

Please note that with particularly large sets of query sequences, this process may take a while, although it does take advantage of multithreading if passed via the *–threads* argument.

BLAST options:

--blastn **BLASTN binary**: Path to blastn binary. If installed via conda install, the bioconda recipe will be used, and placed into the bin/ folder for your conda installation. mrbait, by default, assumes that the blastn executable is locatable in your $PATH as "blastn". If this is not the case, provide an alternate path using this flag.

--makedb **MakeBlastDB binary**: Path to makeblastdb binary. If installed via conda install, the bioconda recipe will be used, and placed into the bin/ folder for your conda installation. Mrbait, by default, assumes that the makeblastdb executable is locatable in your $PATH as "makeblastdb". If this is not the case, provide an alternate path using this flag.

--blastdb **Path to BLAST database**: Full path to formatted database to search against

--fastadb **Path to FASTA database**: If database is provided as a FASTA file, and requires building using the makeblastdb command, provide that path here.

--e_value **E-value threshold**: Minimum e-value cutoff to report BLAST hits [default=0.000001]

--gapopen **Gap opening penalty**: Penalty for opening gaps when performing alignment [default=5]

--gapextend **Gap extension penalty**: Penalty for extending gaps when performing alignment [default=2]

--word_size **Word size**: Word size [default=11]

--nodust **Turn off dusting**: Use this flag to turn off the low-complexity filter using by blastn. Depending on what your goals are for BLAST filtering, this may be necessary. Boolean.

--megablast **Use megablast**: Use this flag to switch to the *megablast* algorithm. It is recommended you use this if you are trying to exclusively find nearly identical alignments to a very closely related genome.

CHAPTER 6

Output Files

Final output of baits will be formatted as FASTA and named $out_baits.fasta (where $out is defined using the -o/–out flag). When the *-t/–print_tr* option is in use, targets will also be output as $out_targets.fasta, with an additional field in the header indicating if these targets passed or failed target selection and filtering.

By default, baits are reported with any ambiguity sequences included (e.g. as a consensus sequence) like so:

```
>Locus1_Target4_Bait1
ATGTAATRAGGTATATG......
>Locus1_Target4_Bait2
TATGAATGTCGCGCGAT......
...
...
...
```

If using the *-x/–expand* option, ambiguities will be reported as all combinations, like so:

```
>Locus2_Target4_Bait1.1
ATGTAATAAGGTATATG......
>Locus2_Target4_Bait1.1
ATGTAATGAGGTATATG......
>Locus1_Target4_Bait2.1
TATGAATGTCGCGCGAT......
...
...
...
```

Baits can also be printed as reverse complement. For example, if the *–expand* option was specified, in addition to *–strand both*:

```
>Locus2_Target4_Bait1.1
ATGTAATAAGGTATATG......
>Locus2_Target4_Bait1.1_revcomp
TACATTATTCCATATAC......
>Locus2_Target4_Bait1.1
```

(continues on next page)

(continued from previous page)

```
ATGTAATGAGGTATATG......
>Locus2_Target4_Bait1.1_revcomp
TACATTACTCCATATAC ......
>Locus1_Target4_Bait2.1
TATGAATGTCGCGCGAT......
>Locus1_Target4_Bait2.1_revcomp
ATACTTACAGCGCGCTA......
...
...
```

mrbait will also produce a .sqlite file (e.g. $out.sqlite) which can be used with the –*resume* flag to restart the pipeline at different stages- for example to re-perform bait filtering with different options. This stores the complete database, including all consensus loci parsed from the alignment input files, all targets, and all bait sequences (including those which failed filtering) and can be used independently with your own SQLite queries.

CHAPTER 7

Benchmarking and Hardware Requirements

Use of an HPC or powerful workstation is not necessary, although could speed things up. Testing was performed on a 2014 iMac with a 4-core Intel i7 processor and 32GB of memory, although only a small fraction of this memory was needed.

7.1 Runtime scaling

With a ddRAD dataset sequencing on HiSeq 2500 paired-end with 150bp reads, including 48 individuals and generating 51,931 alignments, the following command was run:

```
mrbait -L wtd_run1.loci -T 4 -c 12 -l 150 -b 60 -K 1.0 -d 200 -F snp=1,10 -s tile=30
```

A total of 46,219 alignments passed filtering, of which 44,808 included a conserved region long enough for target design. 27,102 targets passed filtering (which was performed based on number of flanking SNPs) and were used to design 43,342 baits. Total runtime across 4 threads was 392 seconds.

Parallel processing is implemented where practical, primarily in steps 1 and 2. For step 1 (alignment parsing), it splits alignment files into groups, to be parsed by each daughter process. During parsing, when an entry must be added to the SQLite database, this cannot be performed in parallel, so a database lock is implemented so that commits to the database are queued. However, the decrease in runtime due to parallelization far outweighs this:

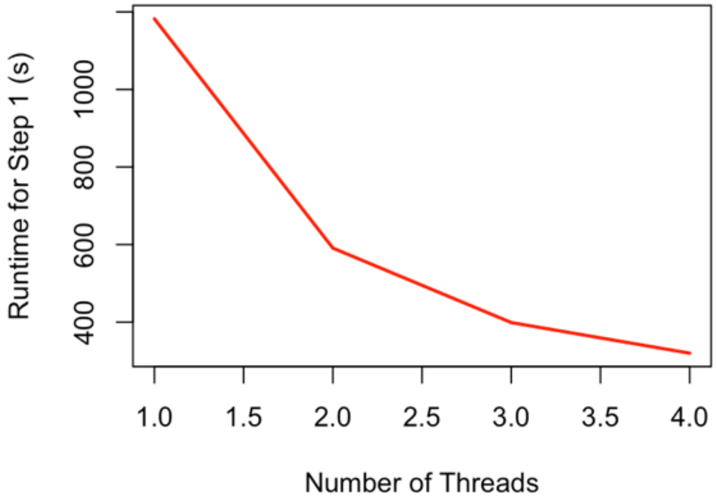


Figure 3: Runtime scaling for Step 1 (most time-intensive step) with varying number of threads

The relationship of runtime to number of threads is similar for step 2 (target discovery), as is the general scheme of preventing database conflicts caused by concurrent database updates. These steps (1 and 2) are by far the most time consuming, although pairwise alignment or BLAST searching in steps 3 or 5 can take considerable time depending on dataset size.

Runtime (in seconds) and peak memory usage (total) for varying numbers of threads, with a ~50k loci RADseq dataset.:

Threads	Step 1(s)	Step 2(s)	Step 3(s)	Step 4(s)	Step 5(s)	Total (s)	Peak mem (MB)	Step 1 mem (MB)
1	1182	129	1	25	3	1342	120	80
2	591	69	1	25	3	690	260	100
3	399	49	1	25	3	478	275	110
4	320	41	1	25	3	392	300	125

7.2 Memory Usage

Large alignments are processed piecemeal, with only a single alignment loaded at a time, thus even large files can be processed without excessive memory requirements. The internal data structure of mrbait relies on a file-based database (SQLite), making it very efficient in terms of memory usage. Because of this, it should run on any normal (and reasonably modern) desktop computer. Peak memory usage tends to be during step 2 (target discovery), as at this step all consensus loci are scattered across threads- note that this also means a slight increase in peak memory requirements as number of threads increases. See Figure 4 for an example of how memory scales throughout the pipeline steps.

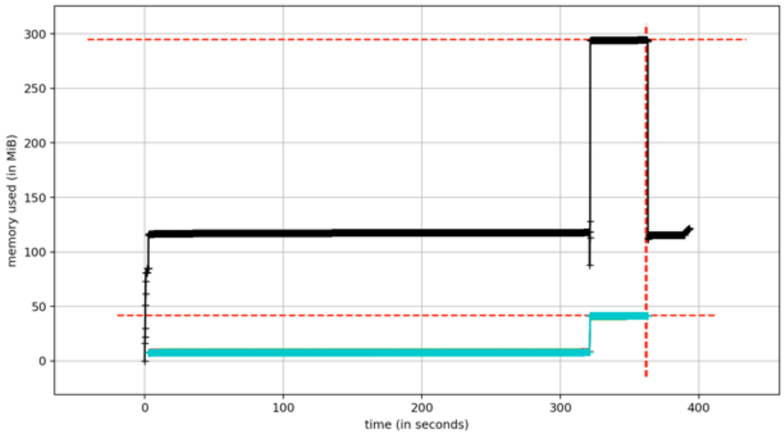


Figure 4: Memory usage throughout the pipeline (recorded for a 50k loci RADseq dataset). Total memory usage (black) shows a peak of ~300Mb during step 2. Per-thread memory usage (blue) cumulatively impacts total memory usage but drops to zero after step 2, as following steps do not utilize parallel computation

CHAPTER 8

Acknowledgements

Computational resources for testing and benchmarking, as well as assembly of RADseq data for testing MrBait, was provided by XSEDE allocations for JetStream: Startup Allocation TG-BIO160058 to Michael E. Douglas, and Research Allocation TG-BIO160065 to Marlis Douglas. Funding to generate RADseq data was provided by by University of Arkansas Endowments (Bruker Professorship in Life Sciences to MRD and 21st Century Chair in Global Climate Change Biology to MED). Thanks are also extended to colleagues at the University of Arkansas: Zach D. Zbinden for contributions to the code base, and Pam L. McDill for lab work generating the test dataset.

We also would like to thank the Editors and 2 anonymous reviewers from Bioinformatics for valuable suggestions to improve the software, its documentation, and the accompanying manuscript submission.

CHAPTER 9

References

Ali,O.A. et al. (2016) RAD Capture (Rapture): Flexible and Efficient Sequence-Based Genotyping. Genetics, 202, 389–400. Baird,N.A. et al. (2008) Rapid SNP Discovery and Genetic Mapping Using Sequenced RAD Markers. PLoS One, 3, e3376. Faircloth,B.C. (2017) Identifying conserved genomic elements and designing universal bait sets to enrich them. Methods Ecol. Evol. Hoffberg,S.L. et al. (2016) RADcap: sequence capture of dual-digest RADseq libraries with identifiable duplicates and reduced missing data. Mol. Ecol. Resour., 16, 1264–1278. Lemmon,A.R. et al. (2012) Anchored hybrid enrichment for massively high-throughput phylogenomics. Syst. Biol., 61, 727–744. McCormack,J.E. et al. (2012) Ultraconserved Elements Are Novel Phylogenomic Markers that Resolve Placental Mammal Phylogeny when Combined with Species Tree Analysis. Genome Res., 22, 746–754. Peterson,B.K. et al. (2012) Double Digest RADseq: An Inexpensive Method for De Novo SNP Discovery and Genotyping in Model and Non-Model Species. PLoS One, 7, e37135.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search