The BaitFisher package manual

BaitFisher version 1.2.7
BaitFilter version 1.0.6

August 2017

Christoph Mayer and Manuela Sann
Forschungsmuseum Alexander Koenig, Bonn, Germany

I About the BaitFisher package

The BaitFisher package consists of two programs: BaitFisher and BaitFilter.

BaitFisher has been designed to construct hybrid enrichment baits from multiple sequence alignments (MSAs) or annotated features within MSAs. The main goal of BaitFisher is to automatically adapt the number of necessary baits to the variability of the underlying alignment. In conserved regions of the MSA, BaitFisher designs fewer baits, in more variable regions it designs more baits. This approach makes use of the fact that hybrid enrichment baits can differ to some extends from the target region, which they should hybridise agains in order to capture and enrich them. By specifying a maximum allowed distance between the designed baits and the sequences in the MSAs the user can control the allowed bait-to-target distance and the degree of reduction in the number of baits that are designed. See the BaitFisher paper Mayer, Sann et al. 2016 for details.

BaitFilter was designed to post-process the baits that have been designed with BaitFisher. BaitFilter can be invoked in multiple steps to conduct one of the following tasks: (i) Determine whether baits bind unspecific to a reference genome, i.e. determined whether the bait is similar to more than one region in the reference genome. (ii) Filter baits that only have partial length matches to a reference genome. (iii) Determine the optimal bait region in a MSA. The optimal bait region can be the most conserved region in the MSA or the region with the highest number of sequences without gaps or ambiguous nucleotides in the alignment. (iv) Convert the (filtered) set of baits into a format which can be uploaded at the bait producing company.

Since performance was one of the major design goals, the BaitFisher and BaitFilter programs are both written in C++.

When using BaitFisher please cite:

Mayer, C., Sann, M., Donath, A., Meixner, M., Podsiadlowski, L., Peters, R.S., Petersen, M., Meusemann, K., Liere, K., Wgele, J.-W., Misof, B., Bleidorn, C., Ohl, M., Niehuis, O., 2016. BaitFisher: A Software Package for Multispecies Target DNA Enrichment Probe Design. *Mol. Biol. Evol.* 33, 1875 - 1886. doi:10.1093/molbev/msw056.

II Where to obtain the BaitFisher package

A Debian package with BaitFisher will be available soon. This will allow you to install BaitFisher on Debian Linux directly from you Linux package manager.

The source code for the BaitFisher package can be downloaded from:
https://github.com/cmayer/BaitFisher-package.git

When downloading BaitFisher from github, we recommend to download the latest release version: *https://github.com/cmayer/BaitFisher-package/releases* or the latest development version: *https://github.com/cmayer/BaitFisher-package/archive/master.zip*.

III Compiling and installing BaitFisher and BaitFilter from source

System requirements:

BaitFisher can be compiled on all platforms, which provide a C++ compiler that includes the following system header files: unistd.h, sys/stat.h, sys/types.h, dirent.h. These header files are standard header files on all unix systems such as Linux and Mac Os X. I also managed to compile BaitFisher and BaitFilter on Windows using the Mingw compiler (www.mingw.org).

Compiling BaitFisher and BaitFilter:

Please, unpack the BaitFisher archive you downloaded.

On the command line change to the BaitFisher-package-master directory. Type `make` on the command line. This compiles the BaitFisher as well as the BaitFilter program. The executables of these to programs are called: BaitFisher-vxxx and BaitFilter-yyy, where xxx and yyy are version numbers of the two programs in the release you downloaded. The binary executables can be copied on your computer to any place you like. To use these executables, they have to be addressed with their path, or they have to be placed in a directory which is included in your systems path variable.

IV Terms you should be familiar with when reading the manual

Multiple sequence alignment (MSA): In this context an MSA will always be an alignment of multiple nucleotide sequences. Baits can only be designed for nucleotide alignments and not for amino acid alignments.

Target MSAs: MSAs for which baits shall be designed.

Target region: DNA region of interest in the genome.

Bait: An artificially assembled RNA sequence that binds (i.e. hybridises) to parts of the target region. It can be used to capture sheared genomic DNA fragments. A bait is able to hybridise agains sequence fragments which are identical or almost identical to its reverse complement.

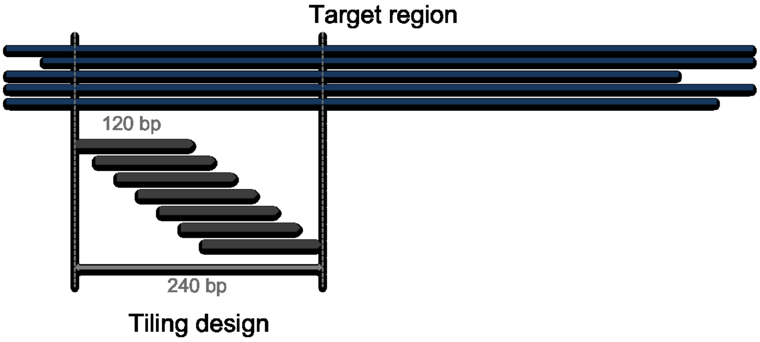

Figure 1: Target region and tiling design. In this example the tiling design consists of seven successive 120-bp bait windows. The complete tiling design spans 240 bp. Starting coordinates of the baits are separated by an offset of 20 bp. A region for which baits have been constructed according to a given tiling design shall be referred to as a bait region.

Usually, a difference of 15% to 20% in the nucleotides between bait and target are not a problem and the bait still binds to the target. (See Mayer, Sann et al. 2016 for details.)

Tiling design: The probability of capturing a given locus increases with the number of baits that are placed into the locus. For this reason, it is common practice not only to use a single bait per locus but a so called tiling design consisting of multiple baits.

With a tiling design one specifies how successive baits are arranged and how many baits are used at a single locus (see Figure 1). A tiling design is completely defined by the following information: the bait length, the number of successive baits, the offset between starting coordinates.

Bait region: A genomic DNA region for which baits have been successfully determined for a given tiling design. The length of the bait region is identical to the length of the tiling design (see Figure 1). While a tiling design is only a description of how baits will be constructed, a bait region is a genomic locus at which designing baits for a full tiling design was successful.

Bait window and bait tiling stack sets: For a given locus of length of the tiling design, BaitFisher will cut the MSA into N bait windows, where N is the number of baits in the tiling design. The first bait window starts at the beginning of the the locus. The second bait window starts with on offset specified in the tiling design, e.g. 20bp in the example above. The starting points of all further bait windows are separated from its successor by the same offset. In order to design Baits for a full tiling design, BaitFisher will design baits for each of the N bait windows. The collection of all baits in one bait windows is called a tiling stack set of baits, or short a "tiling stack". The number of bait windows and the number of tiling stacks are both equal to the number of successive baits in the tiling design.

The number of baits that BaitFisher designs for a given bait window and therefore for a given tiling stack varies and will depend on the similarity of the sequences in the bait window. If the sequences in the bait window are highly similar, the number of required baits will be small. In the extreme case, one bait can be sufficient to represent all sequences in a bait window. If the sequences are very dissimilar, the number of required baits will be high. In the extreme case, the number of required baits will be equal to the number of sequences in the bait window.

In the above example we have 7 bait windows and 7 tiling stack sets of baits in each bait region.

Continuous vs. non continuous sequence data: When designing baits for genomic loci, the underlying assumption is that the sequences in the MSA are continuous, i.e. the sequences in the MSA of the target region are highly similar in full length to the DNA in the genomes of the organisms for which DNA shall be enriched. The sequences in the MSA are called non continuous, if sequence segments have been removed or inserted with respect to the homologous sequence in the genome of interest.

There are important cases in which sequence data might be non continuous: E.g. for MSAs of transcriptomic sequences, it is not guaranteed that the sequences are continuous in the genome, since the splicing machinery might have already altered the original sequences by removing introns. If introns have been spliced out in the transcript sequences, the MSA of transcript sequences contains exon-exon boundaries, which do not exist in the genome since introns separate the exons. This can be important when designing baits. Baits which span the artificial exon-exon boundary, are expected to be less efficient or even worthless compared to other baits.

The BaitFisher package implements two strategies to deal with this problem. (i) BaitFisher implements a method to cut transcript sequences into exons or CDS regions with the help of an annotated genome. (ii) BaitFilter can be used to search for baits which do not match in full length or almost full length to a reference genome. More specifically, BaitFilter can be asked to remove bait regions in which tiling stack exist in which not at least one bait exists which matches with at least a specified proportion of its length with the reference sequence. For more details see the **--blast-min-hit-coverage-of-baits-in-tiling-stack** option, which can be invoked together with one of the following modes: blast-l, blast-c, blast-f, blast-a.

Reference genome: If alignment cutting is requested, an annotated genome is required. See Section VI.2 for details. A genome is also required if a blast search of the baits against a genome is requested. This genome will be called a reference genome, since it serves as a reference either for the features we want to excise or as a reference for for the blast search.

V The algorithm behind BaitFisher

The algorithm behind BaitFisher is described in detail in Mayer, Sann et al. 2016.

VI BaitFishers two major modes: full MSA mode and MSA cutting mode

BaitFisher can design baits for MSAs.

Furthermore, BaitFisher can be used to excise annotated features from MSAs with the aid of an annotated genome. E.g. it can excise exons or CDS regions from transcript alignments.

VI.1 Full MSA mode

This is the standard mode. The user simply specifies the folder containing the multiple sequence alignment files and the tiling design in a configuration file and BaitFisher will design baits for all alignment files and for every possible starting point of a bait region. In this mode the user can e.g. design baits for COI or 18S sequences or for genes excised by the user.

VI.2 Alignment cutting mode

BaitFisher can excise features such as exons or CDS regions from given alignments using an annotation file. The most important example is to excise exons or CDS regions from transcript alignments.

The main reason the user might want to excise e.g. exons or CDS regions from transcript alignments is that transcript alignments might not contain continuous DNA. A transcript alignment might or might not contain introns. If the introns have already been removed by the splicing machinery of the organism the transcripts originate from, then the DNA is not continuous any more. If baits are designed across an exon-exon boundary in the transcript file in which an intron is missing, then there is no such locus in the genome. As a consequence, the efficiency of the baits spanning this exon-exon boundary are expected to be reduced since they cannot bind in full length to the target. In this case one might prefer to design baits only for annotated exons or CDS regions and not for the whole alignment. With this approach, we ensure to design baits for sequence segments which are continuous in the genome.

Alignment cutting can only be performed, if the user has an annotated reference genome in form of a fasta file, containing the genome sequence assembly and a gff file, containing the annotation. Moreover, each MSA for which baits have to be designed, has to have a sequence that stems from the reference genome. The name of this sequence has to encode the taxon name as well as the gene name in the annotation.

The name of the reference species, which is used in the MSA, has to be specified in the BaitFisher configuration file.

If alignment cutting is requested, BaitFisher conducts the following steps:

- For each MSA (usually a gene), BaitFisher searches for the corresponding annotation in the gff file. This is only possible if the name of the gene is encoded in the sequence name of a sequence in the MSA which belongs to the reference genome.
- With the genomic coordinates of the gene features that BaitFisher finds in the gff file it excises the features from the genome and
- aligns them to the transcript sequence that comes from the reference genome.
- The alignment coordinates at which the genomic feature starts and ends within the reference sequence allow to determine the feature coordinates in the MSA.
- The feature can now be excised and BaitFisher will confine the bait design to the excised features.

An example analysis comes with the BaitFisher package. See Section (XI.2) for details.

More detailed description of the alignment cutting procedure:
For being able to cut the alignments into features, BaitFisher requires the following: First, each alignment is only allowed to contain a single gene. The alignment has to contain the sequence of the reference species and the name of the gene has to be encoded in the sequence name of this reference species. The gene name is required to locate the annotation of this gene in the

gff file. Since the sequence names in the fasta file contain several pieces of information, it must be possible to split the sequence names into these pieces with a specific delimitation character. After splitting the sequence name with the delimitation character, there has to be one field that contains the name of the gene written exactly in the same way as the gene is called in the gff file. Another field has to contain the taxon name. The taxon name is required (i) to identify the reference sequence and (ii) to identify the taxon name if required taxa are specified (see below). The delimitation character and field numbers of gene and taxon name can be specified in the parameter file. After extracting the gene name from the sequence name of the reference species, BaitFisher uses the gene name to find all features that belong to this gene. In the gff file each feature is described on a single line. The last tab delimited field in a gff file is the so call attribute field. Usually, the name of the gene is given as an attribute of the form "Parent = ¡GeneID¿", where the "=" symbol can be left out. The keyword parent is often used to indicate the name of the next outer element a feature belongs to, which in this case is the gene. Other keywords are possible. The keyword under which BaitFisher can find the geneID can be specified in the parameter file. The geneID has to match exactly the geneID found in the sequence name of the MSA. Only then will BaitFisher be able to find all features that belong to the geneID of the MSA BaitFisher works on. After BaitFisher determined all features of a specified geneID, it will filter those features who's feature name match the name specified in the parameter file. Typical feature names for which baits are constructed are "CDS" and "exon". After BaitFisher successfully identified features for the MSA it is working on, it uses the gff file coordinates to extract the feature sequences from the specified genome. Each of these sequence segments are aligned to the reference sequence in the MSA. The coordinates at which the feature begins and ends in the transcript sequence will allow BaitFisher to identify the coordinates of the feature in the MSA. In the next step BaitFisher will extract the feature from the MSA. It will save the feature in a separate fasta file for future reference for the user and it will use it to design baits.

No baits will be designed if BaitFisher cannot find a sequence name that encodes the correct name of the reference taxon or if no annotated features can be found in the gff file for this specific gene.

VII Special mode – required taxa

BaitFisher can only design baits for sequences that do not contains gaps or ambiguous characters in the bait window. In a bait windows in which some sequences do not fulfil this requirement, BaitFisher will only include sequences in the bait design process which have defined bases in the whole window. Sequences which do not fulfil this requirement will simply not be considered in the bait design of this particular bait window.

This is an acceptable approach, as long as the missing sequences are not highly important. If, however, the missing sequences are the most important relatives of those species for which baits are designed, one might not want to design baits for regions in which no sequence information is present for the most important species.

If the user considers specific sequences in the MSAs to be of particular importance she/he can specify the corresponding taxon names as required in the parameter file. If one of the required taxa is missing from an MSA or if its sequence contains gaps or ambiguous characters in the bait window, BaitFisher will not design baits for the region of the MSA that contains the missing information.

If required taxa are specified, the sequence names in all MSAs have to contain taxon names, in order to be able to identify the sequences. BaitFisher identifies taxon names by splitting sequence names according to the `sequence-name-field-delimiter}` specified in the parameter file and extracting the taxon name in the field specified by the `sequence-name-taxon-field_number}`option.

The required taxon string has the following format: Groups of taxa of which at least one representative has to be present are enclosed in parentheses. Taxon names within one group are separated by commas. Multiple groups, separated by comma, can be specified.

Example: (Taxon1, Taxon2, Taxon3),(Taxon4),(Taxon5, Taxon6)

In this example a bait window is only valid if either Taxon1, Taxon2 or Taxon3 are present, if Taxon4 is present and finally, if either Taxon5 or Taxon6 are present in full length of the bait window.

VIII Program options, the configuration file and input file formats

In order to design baits for one or multiple MSAs, BaitFisher requires several pieces of information:

i. Details about the tiling design, i.e. the bait length, the number of baits per tiling design, and the offset between the starting coordinates

ii. Directory that contains all input files

iii. Directory into which output files shall be written

iv. Whether or not the alignments shall be cut according to an annotated reference genome.

v. (Optional) If the alignment shall be cut, BaitFisher requires following information:

   (a) The path and filename of the reference genome and the annotation file.

   (b) The name of the feature (spelled exactly as in the gff file e.g CDS) for which baits shall be constructed.

   (c) The name of the reference species with associated sequence in the MSA and related reference genome.

   (d) The name of the attribute field in the gff file that contains the gene name (usually one of: parent, parent-ID, spelled exactly like in the gff file).

   (e) A character (field separator) by which the sequence names in the MSA can be separated to extract the taxon name and the gene name. This is required first, to find the sequence that corresponds to the reference genome and second, to find for each gene MSA the corresponding annotation in the gff file.

   (f) The position at which the taxon name is found in the fasta sequence name after splitting the name according to the field separator.

   (g) The position at which the gene name is found in the fasta sequence name after splitting the name according to the field separator.

vi. (Optional) A string that contains required taxa. Baits will only be constructed for potential bait regions in which all required taxa are present.

BaitFisher program options:

The BaitFisher parameter file is the place where all program options, i.e. the above information has to be specified. The parameter file has to be a plain text file. A word document or other non-pure text formats cannot be read by BaitFisher. The parameter file is allowed to contain "comments". Comments are indicated by the "#" symbol. More specifically, when reading the parameter file, BaitFisher ignores in each line all text after the occurrence of the first "#" symbol. The text after the first "#" can be used by the user to add his own comments. The order in which the required or optional options are specified is irrelevant. A fully specified parameter file can be found in the example analysis.

VIII.1 Required options:

directory-transcript-files

Example: `directory-transcript-files /Users/myname/data/baits/nt-data`

The path to the directory that contains all input MSA files. The path can be an absolute path or relative to the current working directory. All input files have to be aligned sequence files in the fasta format.

bait-length

Example: `bait-length 120`

The length of the baits that are constructed. A commonly used value is 120 bp. Some bait producing suppliers only produce baits of specific lengths, e.g. 120 bp.

cluster-threshold

Example : `cluster-threshold 0.15`

In the process of clustering similar baits, BaitFisher creates cluster with a maximum sequence distance specified by the cluster-threshold. The bait sequence will be determined as an artificial sequence in the centre of each cluster. So the maximum distance of the baits to the sequences in the MSA should be roughly half the value specified by this option.

Typical values are 0.1 to 0.2, which corresponds to 10%-20% sequence dissimilarly within clusters of sequences. If baits are constructed for target sequences very similar to the sequences in the MSAs, this parameter controls that bait-to-target distance.

bait-offset

Example: `bait-offset 20`

The offset between starting coordinates of baits in tiling design. A typical value is 20.

bait-number

Example: `bait-number 7`

The number of baits in the tiling design. Typical values are between 1 and 7.

output-directory

Example: `output-directory /Users/myname/data/baits/bait-results`

Path to the directory BaitFisher will use for output files. The path can be absolute or relative to the current working directory. BaitFisher will write to this directory the loci- baits.txt (see Section XX for more information). If alignment cutting is requested, BaitFisher will also write all excised alignments to this directory in fasta format. This directory has to be created by the user before starting BaitFisher.

VIII.2 Optional options:

required-taxonomic-groups-string

Example:

`required-taxonomic-groups-string (Taxon1, Taxon2),(Taxon3)`

If the user wants to construct baits only for loci at which the MSA has full sequence information for specific taxa, they can be specified here. See Section VI for more information.

sequence-name-field-delimiter

> Example: `sequence-name-field-delimiter |`
>
> If the user specifies required taxon names or if the alignment has to be cut according to annotated features, the taxon, and for alignment cutting also the gene name, has to be specified in each sequence name in the input fasta file. More specifically, the sequence name has to be separated by the sequence-name-field-delimiter into different parts. One of these parts must be the taxon name, if alignment cutting is requested, another part must contain the gene name. This option must be a single character. This parameter is ignored if no required taxa are specified and no alignment cutting is requested.

sequence-name-taxon-field_number

> Example: `sequence-name-taxon-field_number 2`
>
> If either a required taxon string is specified, or if alignment cutting is requested, BaitFisher will divide all sequence names of input files according to the sequence-name-field-delimiter. This option specifies the index of the field that contains the taxon name. The first index is index 1. This parameter is ignored if no required taxa are specified and no alignment cutting is requested. Default value: 2

sequence-name-geneID-field_number

> Example: `sequence-name-geneID-field_number 3`
>
> If alignment cutting is requested, BaitFisher will divide all sequence names of input files according to the sequence-name-field-delimiter. This option specifies the index of the field that contains the gene name. The first index is index 1. This parameter is ignored if no alignment cutting is requested. Default value: 3

Hamming-center-sequence-computation-mode

Example: `Hamming-center-sequence-computation-mode heuristic`

BaitFisher implements an exact and a heuristic mode to compute the one-center sequence. The exact mode is slightly better in finding the artificial sequence that minimises the maximum distance to all sequences in a cluster. For a large number of sequences in a cluster the computation time can be enormous, so that only the heuristic mode can be recommended. The heuristic mode is much faster and only slightly less accurate. Allowed values: exact, heuristic. Default: heuristic.

verbosity

Example: `verbosity 3`

The integer value specifies how much runtime output BaitFisher writes to the terminal (i.e. to standard output and standard error). If this value is set to 0 BaitFisher runs in silent mode and only informs about critical errors that force it to exit. With a value of 1, BaitFisher also writes warning to the the standard output. With a value of 2 BaitFisher provides regular progress messages. All higher values are mainly for debugging purposes. The output reaches a maximum level at a value of 10000. Default: 5.

Options that have to be specified if alignment cutting was set to yes. If no alignment cutting is requested, these options are ignored.

reference-genome-file

Example:

```
reference-genome-file /Users/myname/data/ref-genome/Apis.fas
```

The filename of the fasta file that contains the genome assembly of the reference genome. This option is only required if alignment cutting is set to yes.

gff-file-reference-genome

Example:

```
gff-file-reference-genome /Users/myname/data/ref-genome/Apis.gff
```

Name of the gff file with annotation information for the reference genome specified by the reference-genome-file option. This option is only required if alignment cutting is set to yes.

reference-genome-name

Example: `reference-genome-name Apis_mellifera`

Taxon name of the reference genome. In order to be able to cut out features, BaitFisher has to find a sequence with this taxon name in each MSA it has been given to designs baits. MSAs which have no sequence that contains this taxon name in the taxon name field of its sequence name cannot be processed. A warning will written to the standard output, if no sequence with this taxon name is found and if the verbosity ≥ 2.

Options that should be specified if alignment cutting was set to yes. If no alignment cutting is requested, these options are ignored.

gff-feature-name

Example: `gff-feature-name CDS`

Name of the gff feature that shall be used for alignment cutting. Typical names are: CDS, exon. Default: CDS. This name has to be spelled and capitalised exactly as the name of the feature in the gff file.

gff-record-attribute-name-for-geneID

Example: `gff-record-attribute-name-for-geneID Parent`

Name of the gff attribute that specifies the geneID. Each line in the gff file consist of 7/9 tab delimited fields. The last field contains any number of attributes. For features such as CDS, exon, introns, etc, the attribute string has to contains an attribute with the geneID. Typically the name of this attribute is Parent. Default: parent. Apart from capitalisation of this name, the name as to be identical to the one used in the gff file. The geneID found in the gff file has to match the geneID found in the gene name field of the input sequences.

remove-reference-sequence

Example: `remove-reference-sequence yes`

Allowed values: yes or no. If the sequences from the reference genome are added to the MSAs only for the purpose to allow the alignment to be cut into its features, it is usually not intended by the user to design baits for the reference sequence. In this case the reference sequence should be removed. If the reference sequence belongs naturally to the sequences for which baits are designed, it should not be removed.

IX Starting a BaitFisher analysis on the command line

If the BaitFisher program has not been placed into one of the directories included in the system path, it has to be called with the full path to where the binary file is located. If it is placed in the system path, it can simply be called with its name. The only but required command line parameter is the name of the parameter file.

BaitFisher-v1.2.7 name-of-parameter-file

Make sure that the output directory specified in the parameter file exists, since it will not be created automatically.

X The output format

In the directory specified in the parameter file, BaitFisher writes a bait file with the standard name: baits-loci.txt. This file contains one line of output for each bait region for which baits could be designed successfully.

For each bait region BaitFisher produces the following output on one line. Column numbers refer to the tab delimited fields the line of output.

1_{st} column: Name of alignment file this bait region belongs to.

2_{nd} column: Gene name in case of alignment cutting, or the name of the alignment again.

3_{rd} column: Feature number of requested kind (e.g. CDS) in case of alignment cutting, or 0 otherwise.

4_{th} column: Maximum number of features of requested kind (e.g. CDS) in case of alignment cutting, or 0 otherwise.

5_{th} column: Starting coordinate of bait region in the alignment

6_{th} column: Number of sequences which where fully available in the bait region. Note: If you requested a tiling design with just one bait per bait region, the maximum number can be the number of sequences. If the number is lower than the number of sequences, some sequences contain gaps or ambiguous characters in this bait region. If a tiling design was requested that has more successive baits per bait region, say N, the number of sequences which where available will have an upper limit of N*number_of_ sequences_in_the_MSA. Again, the maximum value N*number_of_sequences_in_the_MSA will be reached if all sequences have no gaps or ambiguous characters.

This column can be interpreted as the amount of evidence the baits in the bait region are based on. If a large alignment contains only one valid sequence in a certain region of the MSA, BaitFisher will still construct baits for this region, unless required taxa are missing. After bait design, regions with insufficient information content for the bait design can only be spotted by looking for suspiciously small numbers of sequences in this column.

7_{th} column: Number of baits required for the full tiling design in this bait region. An upper limit for this number is the number of sequences from column 6. The greater the difference is between column 7 and 6, the more the user can benefit from clustering sequences and computing only one center sequence per cluster.

8_{th} column: Baits for this bait region. This last field describing the bait region contains several bits of information: For each bait is first shows the “stack number”. For a tiling design with N successive baits we have N successive bait windows, in the following called bait stacks. The numbers are integer values in the range 1 ... N. Followed by the stack number comes the bait sequence. After each bait sequence there are two real numbers: The CG content of the bait and the maximum distance of the bait to the sequences in its cluster. Multiple baits are separated by commas. The number of baits that will be found per line is equal to the number specified in column 7.

Obviously, with one bait region per possible starting position, BaitFisher produces a highly redundant output. The reason is that some starting positions might be more efficient in terms of “required number of baits” or “number of sequences the result is based on”. The redundant output allows the user to choose for each locus of interest the optimal bait region. This topic is addressed in Section XI.

XI Example analyses

The BaitFisher package includes two example analyses, one with and one without alignment cutting. The example with alignment cutting is also an example in which a required taxa string has been specified. Both examples are included in the BaitFisher package, in the Example folder. The example with alignment cutting requires a full genome as well as a gff file. Since they are too large, we included reduced files and provide download links for the original files.

XI.1 Example without alignment cutting:

In order to run this example analysis, compile and install BaitFisher as described above. On the command line, change to the Example/Example-without-alignment-cutting folder. In this folder you find the parameter.txt file which contains all BaitFisher options for this example run. In the alignment folder you find three gene MSAs for which baits shall be designed. In order to run the analysis type run-example.sh on the command line. This requires that the BaitFisher program can be found in the relative path ../../BaitFisher-v1.2.7. If the executable has been moved somewhere else, either modify the command in the run-example.sh file or recreate a copy of the executable in the main BaitFisher folder.

The example analysis should finish within a few seconds.

This run produces a single output file in the folder BaitFisher-results. The first few lines in this output file, which result from processing the input file EOG5DV4JW.fas look as follows:

EOG5DV4JW.fas EOG5DV4JW.fas 0 0 1 173 19 1 ATG . . .

EOG5DV4JW.fas EOG5DV4JW.fas 0 0 2 173 18 1 TGG . . .

EOG5DV4JW.fas EOG5DV4JW.fas 0 0 3 173 19 1 GGC . . .

EOG5DV4JW.fas EOG5DV4JW.fas 0 0 4 173 19 1 GCT . . . . . .

A full description of the tab-delimited columns is given above. Since the columns 3 and 4 only play a role if alignment cutting is performed, the columns are filled with zeros in this example. Column 5 contains the starting position of the bait region in the alignment. We see that BaitFisher computes baits for each possible starting position of the bait region. Column 6 is the number of complete sequences the baits in this bait region are based on. We observe that all starting positions have the same sequence coverage in the bait region. (See Section **??** for details.) Column 7 shows the number of required baits for this bait region. The smallest number of required baits is found for starting position 2. In this example we have chosen a rather conserved gene, with the result that BaitFisher can drastically reduce the number of baits required in comparison to the number of baits required if baits are constructed for each sequence. The proportion of baits we can save here is is: (1-18/173)*100% = 89.6%.

XI.2 Example with alignment cutting:

On the command line, change to the directory
`Example/Example-with-alignment-cutting-and-required-taxa`.

In principle a BaitFisher analysis with alignment cutting requires a whole genome and the corresponding annotation file. Since we will only analyse five transcript files containing five genes in this example, we reduced the whole genome file to the scaffolds required in this example and the corresponding annotation file to the annotation corresponding to the genes in this example. This reduction allows us to make this example to be included in the git-hub repository.

The reference genome we use in this example is the *Nasinia vitripennis* genome. The reduced genome can be found in the Nvit-genome folder of this example. The reduced genome is a modified version of the whole genome which can be downloaded together with the annotation file here [1].

The annotation file has to be in the gff-format and has to correspond to the fasta file. To get an idea how the annotation file looks like the first few lines are shown:

```
SCAFFOLD4 OGSv12_RefSeq transcript 3975063 3977809 . - . GenePrediction NV11525-RA
SCAFFOLD4 OGSv12_RefSeq exon   3975063 3976592 . - . GenePrediction NV11525-RA
SCAFFOLD4 OGSv12_RefSeq exon   3977689 3977809 . - . GenePrediction NV11525-RA
SCAFFOLD4 OGSv12_RefSeq mRNA   3975384 3977706 . - . GenePrediction NV11525-RA
SCAFFOLD4 OGSv12_RefSeq CDS    3975384 3976592 . - 0 GenePrediction NV11525-RA
SCAFFOLD4 OGSv12_RefSeq CDS    3977689 3977706 . - 0 GenePrediction NV11525-RA
```

This gff-file is always tab delimited and contains nine columns. The last column can contain several key/value pairs. A key and its value can be separated by a space or the "=" symbol. Several key value pairs are separated by semi-colons. In this example the last and nineth column consists of a single key-value pair given by the string "GenePrediction gene-name", where gene-name is the name of the gene in the official gene set. Note that in the gff-file extract above the tabs between the columns and the space in the last column cannot be distinguished.

In brief:
There are two important correspondences between the gff file and the remaining data. Both have to be fulfilled to make alignment cutting into features possible.

(i) The sequence names in the genomic fasta file have to correspond to the names given in the first column of the gff file. Furthermore, the coordinates in the fourth and fifth column of the gff file have to correspond to the coordinates of the gene in the fasta file. This correspondence should be fulfilled if the two files are obtained from the same source and are based on the same assembly and annotation. Annotation versions and assembly versions can differ such as in the case of the Nasonia genome.

(ii) The second correspondence is a bit more tricky. The gene name found in the gff file, in the above example this is *NV11525-RA*, has to be encoded in the sequence name of the reference species in the transcript files.

In detail:
While the correspondence (i) is usually fulfilled if the sequences are downloaded from the same source, the second correspondence needs more explanation:

BaitFisher has to know which transcript or gene files corresponds to which gene names in the gff file. In order to make this connection, the transcript files have to contain a sequence originating from the reference species, in our example from *Nasonia*. This sequence has to have a sequence name in which the gene name is encoded in a way, so that it can be extracted by BaitFisher.

---

[1] `http://www.hymenopteragenome.org/drupal/sites/hymenopteragenome.org.nasonia/files/data/Nvit_1.0_scaffolds.fa.gz` and `http://hymenopteragenome.org/nasonia/nasonia_genome_consortium/data/Nvit_OGSv1.2.gff2.gz`

Let us give an example: One of the transcript files in the alignments folder is the file: `EOG5B2RVS.linsi.nt_without_bees_ants_rmgo.fas`.

This looks as follows:

```
>EOG5B2RVS|HSALT_3.3|Nysson_niger|C277098-531
ATG...
>EOG5B2RVS|AECHI_3.8|Dinetus_pictus|scaffold1253-531
ATG...
...
>EOG5B2RVS|NVITR_1.2|Nasonia_vitripennis|NV16710-RA
ATG...
>EOG5B2RVS|LHUMI|Sphecius_convallis|C324842-531
ATG...
>EOG5B2RVS|LHUMI|Stizoides_tridentatus|C357502-531
ATG...
>EOG5B2RVS|AECHI_3.8|Psenulus_fuscipennis|C384528-531
ATG...
```

Note that each sequence name is delimited by the "|" character into 4 fields. Important fields are the third and fourth field in this example: The third field contains the species name and the fourth field contains the gene name for species for which a whole genome is available.

Alignment cutting is possible if one of the sequences has the name of the reference species in the third field. In our example this is the name `Nasonia_vitripennis`. This sequence name has to have a gene name in the fourth field which corresponds to one of the genes found in the gff file. In this example this is the gene name `NV16710-RA`.

Without these correspondences, BaitFisher would not know how to find the features in the gene files and how to excise them. Further down we will discuss some approaches which allow us to obtain transcript files with the appropriate sequence names.

For the moment we assume we have corresponding transcript or gene files, a reference genome and the annotation file and we will have a look at the parameters which have to be set in the parameter file. The following options specified in the parameter file of this example analysis are all directly related to the alignment cutting:

This option switches alignment cutting on:

```
alignment-cutting yes
```

A sequence of the reference species has to be included in every transcript or gene alignment. The reference species might not be of further interest for the project and one might not want to design baits for the reference species. In this case one can tell BaitFisher to remove the reference species from the alignment before the bait design procedure. If the reference species shall be removed, BaitFisher will also remove gap only columns from the alignment, which can arise if the presence of the reference species lead to gaps in all other sequences. Removing gap only columns can change to coordinates in the transcript or gene alignments. The user will find the final feature alignments for which baits have been designed in the specified output folder.

With this command, BaitFisher removes the reference species from all transcript alignments.

`remove-reference-sequence yes`.

The use might have transcript alignments in which the sequence name and the gene name are encoded but maybe with a different field delimiter or in different fields.

The following parameters specify the field delimiter and the field numbers in the parameter file:

```
sequence-name-field-delimiter  |    # Default: "|". Must be a single character
sequence-name-taxon-field_number 3    # Default: 2
sequence-name-geneID-field_number 4    # Default: 3
```

GFF files tend to look very different. Different genome projects unfortunately use different names for the same things. BaitFisher e.g. has to find the gene name in the gff file. As described above, this gene name can be found in the ninth column of the gff file, where it occurs as in key value pair. In our example the key is `GenePrediction`. In other projects the key for the gene name can be different. Examples are `name` or `ParentID`. The key word after which the gene name can be found can be specified in the parameter file. In this example:

```
gff-record-attribute-name-for-geneID    GenePrediction
```

The user might want to excise different features. In the gff file, the feature is given in the third column. Reasonable features for our purpose are `exon` or `CDS`, since only these two are generally annotated and are guaranteed to yield sequences that are continuous in the genome. The name of the features that shall be excised can be specified in the parameter file.

```
gff-feature-name CDS
```

Finally, BaitFisher has to know were to find the genome file and the gff file. Furthermore, it has to know the name of the reference species in order to identify the reference sequence in the transcript or gene alignments.

These three parameters are specified in this example as follows:

```
gff-file-reference-genome Nvit-genome/Nvit_OGSv1.2_reduced.gff2
reference-genome-file              Nvit-genome/Nvit_1.0_scaffolds_reduced.fa
reference-genome-name Nasonia_vitripennis
```

The algorithm of BaitFisher can be outlined as follows:
(i) Read every transcript or gene alignment into memory. (ii) Determine the reference sequence by its name. (iii) Determine the gene name specified in the sequence name of the reference species. Look up the gene name in the gff file and extract all features which have this gene name and which are of the requested feature type (e.g. exon). (iv) For each feature found in the gff file in step (iii) excise the genomic nucleotide sequence of the feature of the reference species from the genome file using the sequence name ("SCAFFOLD4" in the above example) and the coordinates of the feature in the genome file. Align the feature sequence that has been excised to the sequence of the reference species in the transcript or gene alignment. The feature should be identical to part of the transcript alignment. The start and end coordinates of the feature in the reference sequence will tell us where the feature starts and ends in the transcript alignment. With these coordinates, BaitFisher excises the feature from the transcript/gene alignment. If the user requested to remove the reference species from the alignments, it will be removed. All gap only columns resulting from this step will be removed as well. If the feature alignment is long enough to host a full bait region, the bait design will be started.

Obtaining transcript or gene alignments with sequence names which encode the species names and the gene names:
The most difficult part in the alignment cutting mode is to obtain files with corresponding names. If transcripts are put together with the Orthograph software, the sequence names should be in the format required by BaitFisher. Otherwise script have to be written to modify in the input files.

XII BaitFilter and how to proceed after obtaining the BaitFisher output file

After BaitFisher has designed baits for all possible starting positions, we have to do some post processing of the output file. This post processing can be done with the BaitFilter program or with custom scripts written by the user.

BaitFilter is a command line program that has the following command line options. Some options are required and cannot be omitted. These options are indicated below.

-h, --help

Displays usage information and exits.

--version

Displays version information and exits.

--, --ignore_rest

Ignores the rest of the labeled arguments following this flag.

-i <string>, --input-bait-file-name <string>

(required) Name of the input bait locus file. This is the bait file

obtained from the BaitFisher program.

-o <string>, --output-bait-file-name <string>

(required) Name of the output bait file. This is the file the

contains the filtered bait loci and the baits.

-c <string>, --convert <string>

> Allows the user to produce the final output file for the bait producing company. In this mode, BaitFilter reads the input bait file and instead of doing a filtering step, it produces a costume bait file that can be uploaded to the baits producing company. In order to avoid confusion, a filtering step cannot be done in the same run as the conversion. If you want to filter a bait file and convert the output, you will need to call this program twice, first to do the filtering and second to do the conversion. Allowed conversion parameters currently are: “four-column-upload”, which is accepted by several companies. The original format name for the “four-column-upload” format was “Agilent-Germany”, which is still a silently accepted format name.

The four-column-upload format contains information potentially of interest for the researcher. The output contains four columns delimited by tabs. The four columns are: TargetID, ProbeID, Sequence, and Replication. The TargetID and the ProbeID are automatically generated and contain information about the bait locus. The information differs slightly in the two possible cases: alignment cutting is used or not used.

Example with alignment cutting:

If alignment cutting has been used, the output file consists of lines in the following format:

```
GB42203-PA_9 Project1_GB42203-PA_9_11_1_1_1 AACAGTTTACGCACTCCTGATTTAACTTGGGAAAAAGTAAGATCTCAAGTGGACCATGTAATCTGGCCAGATGGAAAGCGT
ATCGTCCTCCTAGCAGAAGGTCGTTTGGTCAACTTATCT 1
```

If alignment cutting is used, the TargetID contains the gene name, in this example “GB42203-PA”, followed by `_9`, where the number is number of the feature. In this case it is the ninth CDS region of the GB42203-PA gene.

If a probe ID-prefix is specified when calling BaitFilter, this string is prepended to all probeIDs in the output file. In the above example, the specified probe ID-prefix was `Project1_`. If alignment cutting is used, the probe ID-prefix is followed by the gene name, which in turn is followed by the string `_9_11_1_1_1`. The numbers separated by underscores are:

- the number of the feature the bait is in (here the 9th CDS region),
- the total number of features in this gene (here the gene contains 11 CDS regions),
- the start coordinate within the feature (here it is base 1),
- the tiling index, i.e. the index of the bait window within the tiling design (here it is bait window 1).
- the number of the bait in the bait window (here this is the 1st bait in this bait window).

After the ProbeID comes the bait string, which is followed by the replication number which is always set to one by BaitFilter.

Example without alignment cutting:

If no alignment cutting has been used, the output file consists of lines in the following format:

```
file1.fas_0 Project2_file1.fas_0_0_45_2_3 ATTTTTGCCTTATGAGCAGGTATACTAGGAACCTCTCTCAGAATATTAATCCGATTAGAACTAAGAAATCCTGGATCTTGAATTAA
CAATGATCAAATCTACAATTCCACTATTACTGCT 1
```

In this case, the TargetID is the sequence name followed by `_0`, where the `_0` is appended only for consistency reasons. If the user wants to replace the TargeID with a different string, this has to be done in a text editor.

The ProbeID is composed of the probe ID-prefix, the file name and the string `_0_0_45_2_3`. Since a feature number and the number of features in this alignment is not applicable if no alignment cutting is performed, these two numbers are

0. They are part of the output only for consistency reasons. The remaining numbers have the same meaning as above. In this example, the bait starts at position 45 of the alignment, the bait is in the 2nd bait window of the tiling design. Within this bait window, this is the 3rd bait.

The `ID-prefix` option and its purpose are explained in more detail below.

New output formats can be added upon request. Please contact the author Christoph Mayer, Email: *c.mayer.zfmk@uni-bonn.de*.

-S

Do not filter any baits but only compute stats for the input file. No output file needs to specified and if an output file is specified, it is ignored. In all filter modes this status option in turned on automatically. This option is useful if stats, but no analysis is needed.

-m <string>, --mode <string>

Appart form the input file this is the most essential option. This option specifies which filter mode BaitFilter uses. The following modes are available.

"ab": Retain only the best bait locus for each alignment file (e.g. gene) when using the optimality criterion to minimize the total number of required baits.

"as": Retain only the best bait locus for each alignment file (e.g. gene) when using the optimality criterion to maximize the number of sequences the result is based on.

"fb": Retain only the best bait locus for each feature (e.g. CDS) when using the optimality criterion to minimize the total number of required baits. This mode can only be used if alignment cutting has been used in BaitFisher.

"fs": Retain only the best bait locus for each feature (e.g. CDS) when using the optimality criterion to maximize the number of sequences the result is based on. This mode can only be used if alignment cutting has been used in BaitFisher.

"blast-a": Remove all bait loci of ALIGNMENTs for which one or more baits have multiple good hits to a reference genome.

"blast-f": Remove all bait loci of FEATUREs for which one or more baits have multiple good hits to a reference genome.

"blast-l": Remove bait LOCI that contain a bait that hos multiple good hits to a reference genome.

"blast-c": Conduct a blast of the baits against a reference genome and do a coverage filter of bait regions. Requires the **--blast-min-hit-coverage-of-baits-in-tiling-stack** option to be specified. See the description of the **--blast-min-hit-coverage-of-baits-in-tiling-stack** for details.

"thin-b": Thin out a bait file to every Nth bait region, by determining the start position that minimizes the number of baits.

"thin-s": Thin out a bait file to every Nth bait region, by determining the start position that maximizes the number of sequences.

--blast-second-hit-evalue <real number>

Maximum evalue for the second hit. A bait is characterised to bind ambiguously, if we have two good hits. This option is the e-value threshold for the second hit.

--blast-first-hit-evalue <real number>

Maximum e-value for the first hit. A bait is characterised to bind ambiguously, if we have two good hits. This option is the e-value threshold for the first hit.

--blast-min-hit-coverage-of-baits-in-tiling-stack <real number>

Specifies the minimum hit coverage for the best blast hit of the baits of a tiling stack against a reference genome. If one tiling stack of a bait region fails to have a bait with this minimum hit coverage, the whole bait region is discarded. The hit coverage is defined as the hit length of the blast hit divided by the length of the bait. This option is required in the mode "blast-c". If this option is not specified in the modes blast-l, blast-a, blast-f no hit coverage is checked.

Example: `--blast-min-hit-coverage-of-baits-in-tiling-stack 0.95`

This requires that one bait has to exist in each tiling stack which has a hit to the reference genome where the hit covers 95% of the bait. If the required baits do not exist, the bait region is discarded.

This parameter can only be used in the modes blast-c, blast-l, blast-f and blast-a. Since the order in which the coverage filter and the other filters are applied matters, the order is defined as follows:

For the mode options: blast-a, blast-f, blast-l the hit coverage is checked *after* filtering for baits with multiple good hits to the reference genome. For the mode option blast-c only a coverage filter is being carried out.

--ref-blast-db <string>

Base name to a blast data base file. This name is passed to the blast command. This is the name of the fasta file of your reference genome.

IMPORTANT: The makeblastdb program has to be called before starting the BaitFilter program. makeblastdb takes the fasta file and creates data base files out of it.

-b <string>, --blast-result-file <string>

Conducting a blast analysis of all baits against a reference genome can take a long time. If different filtering parameters, e.g. different coverage thresholds, are to be compared, the same blast analysis has to be done multiple times. With this argument, the blast analysis will be skipped and the specified blast result file will be used. This option has to be used with caution! No checks are done (so far) to ensure that the blast result file corresponds to the specified bait file. If a BaitFilter run was conducted which did a blast search, BaitFilter will not delete the blast result file after the run was completed. The result file with the name `blast_result.txt` will remain in the working directory. It can be moved or renamed and with this option it can be specified as the input file for further BaitFilter runs. If you have the slightest doubt whether you are using the correct blast result file, you should not use this option. This option is only allowed in modes that would normally do a blast search. This option cannot be specified together with the blast-executable, blast-evalue-cutoff, blast-extra-commandline, ref-blast-db options, since these are options specific to runs in which a blast search is conducted.

--blast-extra-commandline <string>

When invoking the blast command, extra command line parameters can be passed to the blast command. As an example with this option it is possible to specify the number of threads the blast command should use.

--blast-evalue-cutoff <real number>

When conducting a blast search, a maximum evalue can be specified when calling the blast program. The effect is that hits with a higher evalue are not reported. BaitFilter always specifies such an evalue when calling the blast program. The default evalue passed by BaitFilter to the blast program is twice the –blast-second-hit-evalue. If a coverage filter is requested the default value is set to 0.001 if twice the value of –blast-second-hit-evalue is smaller than 0.001. This should guarantee that all hits necessary for the blast and/or coverage filter are found. If the user wants to set a different evalue threshold, this can be specified with this option. With version 1.0.6 of this program, the value is automatically changed to be larger or equal to 0.001 if the coverage filter is used. This makes the usage of this option unnecessary in most cases.

-B <string>, --blast-executable <string>

Name of or path+name to the blast executable. Default: blastn. Minimum blast version number: Blast+ 2.2.x

-t <positive integer>, --thinning-step-width <positive integer>

Thin out the bait file by retaining only every Nth bait region. This option specified the step width N. If one of the modes thin-b, thin-s is active, this option is required, otherwise it is not allowed to set this parameter.

--ID-prefix <string>

This option can only be specified in the `-c` or `--convert` mode, in which bait files in the BaitFisher format are converted to files that can be uploaded to bait composing companies.

When uploading bait files to bait producing companies, the ProbeID, i.e. the ID in the second column of the "four-column-upload" format has to be unique, even for different projects of the same user.

Since the gene name and position numbers might be identical in different projects, we introduced this option to allow the user to make ProbeIDs unique over different projects.

With this option the user is able to specify a prefix string for all probe IDs in the output file in the four-column-upload format. This prefix string is prepended to the ProbeID which is automatically generated by BaitFiler.

--verbosity <unsigned integer>

The verbosity option controls the amount of information BaitFilter writes to the console while running. 0: report only error messages that lead to exiting the program. 1: report also warnings, 2: report also progress, 3: report more detailed progress, >10: debug output.10000: maximum possible diagnostic output. Default: 2.

Example applications:

i. We might want to search for baits that bind unspecific. This can be checked with the BaitFilter program, which in turn does a Blast search of all baits against a reference genome. If a bait has two or more good hits to different positions of the genome assembly, and if this is not an assembly artefact, there is some chance that the bait enriches two or more different loci. The user can choose to remove the bait region, the feature, the gene/alignment the bait belongs to. We propose to remove at least the bait-region in case one of its baits has a good chance to bind unspecific.

   The command used to conduct this analysis is the following:

```
BaitFilter-v1.0.6 -i loci_baits.txt
                  --blast-first-hit-evalue 0.00000001
                  --blast-second-hit-evalue 0.00001
                  --ref-blast-db my-blast-db_of_reference_genome
                  -o baits_filtered_by_blast-l
                  -m blast-l
                  --blast-extra-commandline "-num_threads 12"
           1> o_filter_blast.log 2> e_filter_blast.log
```

   This command conducts a blast filter run with mode “blast-l”, i.e. bait regions that have at least one bait with a blast hit e-value smaller than 0.00000001 and a second good blast hit with an e-value smaller than 0.00001 are removed from the bait-file, since the bait is expected not to bind specifically. The blast program must with version 2.2.25+ or higher must be installed in the system path for this to work. If the Blast program is not in the system path, the full path can be specified with the –B option. The user has to invoke makeblastdb to make a blast data-base of the reference genome prior to starting BaitFilter. In this example the console output is redirect to two files. Namely, the standard output is redirect to the file `o_filter_blast.log` and the standard error is redirected to the file `e_filter_blast.log`.

ii. We might want to ensure that at least one bait per tiling stack maps in full length to a given reference genome. This check can be done if on the one hand we are not sure that the MSAs we use for bait design have indeed continuous gDNA and that alignment cutting cannot be done. The hit coverage can only be checked in conjunction with another blast filter mode. See comman line option above.

   Example:

```
BaitFilter-v1.0.6 --blast-min-hit-of-bait 0.85
                  -i loci_baits.txt
                  --blast-first-hit-evalue 0.00000001
                  --blast-second-hit-evalue 0.00001
                  --ref-blast-db my-blast-db_of_reference_genome
                  -o baits_filtered_blas
                  -m blast-l
                  --blast-extra-commandline "-num_threads 12"
           1>  o_filter_blast.log 2>  e_filter_blast.log
```

In this example we apply two filter in one call to the BaitFilter program. BaitFilter first checks for multiple hits. Only bait regions in which all baits bind specifically are passed to the second filter in which baits are checked for a minimum hit coverage.

iii. After applying the above filters, it is likely that we still have multiple bait regions per alignment or per feature and we probably want to reduce the number of bait regions to one per feature or one per gene or alignment. BaitFilter has two optimality criteria: Minimize the number of required baits in the target region or maximize the number of sequences used to design the baits.

Example:

```
BaitFilter-v1.0.6 -m fb -i baits_file -o baits_filtered
     1>  o_BlastFilter 2> e_BlastFilter
```

This command only retains the best bait region per feature using the number of baits as the optimality criterion. As above, the console-output is redirected to two files.

iv. With BaitFilter version 1.0.6 a new filter mode has been introduced. This mode allows to keep not only one but multiple bait regions per feature, or alignment, if the length of the feature or alignment is long enough so that two or more bait regions can be situated in the alignment with a minimum distance.

Example:

Will be added soon.

XIII BaitFilter example analyses

BaitFilter example analyses are available for the two BaitFisher example analyses introduced above.

Example without reference genome:

You can run this example if BaitFilter has been compiled and if the BaitFilter example analysis in the folder Example/Example-without-alignment-cutting has been completed. To conduct the BaitFilter analysis, change on the command line from the Example/Example-without-alignment-cutting folder to the BaitFilter-analysis folder and type ./run-BaitFilter on the command line. This will produce four bait-files and eight log files. The first two output files are “intermediate” bait files. One of these contains the best bait regions when minimizing the number of baits and one contains the best bait regions when maximizing the number of sequences the baits are constructed from. Since we analysed three alignment files, the output files will contain baits for three bait regions, one for each of the alignment files.

The information BaitFilter write to standard error for the analysis finding the bait region with the smallest number of required baits reads:

```
After determining the best bait region per alignment/gene under the criterion to
minimize the number of baits we have
Bait regions have been determined for this number of different:
```

```
Alignment files 3
Feature (is identical to number of files if not applicable: 3
Bait regions: 3
Total number of baits: 187
Total number of sequences at loci: 479
Proportion of baits saved: 60.96%
Total time used: 0:0:2
Your BaitFilter analysis finished successfully.
```

After producing the intermediate bait files, BaitFilter is invoked again to convert the intermediate bait file into a format that can be uploaded at bait producing companies.

BaitFiter again produces 2 log files in each run and the specified output file.

Example with reference genome:

Will be added soon.

XIV FAQ

(1) **After sequencing the enriched library, the sequences will be assembled. This assembly will contain the whole or a part of the bait region. Can the regions be used in further analyses, e.g. phylogenetic analyses, or does it have to be removed in the same way a primer sites has to be removed when using the Sanger sequencing technique?**

Bait regions do not have to be removed from the data set. In the case of primer regions in the Sanger sequencing method, it was more likely that the output sequence contains the exact primer sequence than the potentially different species sequence. Therefore primer regions had to be removed from the data set.
For hybrid enrichment methods, bait regions do not have to be removed. The baits are digested in the lab and no remnants should be sequenced. The resulting sequence should originate 100% from the target species, of no other problems arose. Therefore is impossible that the bait sequence "overwrites" the species sequence in the bait region.

XV References

Mayer, C., Sann, M., Donath, A., Meixner, M., Podsiadlowski, L., Peters, R.S., Petersen, M., Meusemann, K., Liere, K., Wgele, J.-W., Misof, B., Bleidorn, C., Ohl, M., Niehuis, O., 2016. BaitFisher: A Software Package for Multispecies Target DNA Enrichment Probe Design. *Mol. Biol. Evol.* 33, 1875 - 1886. doi:10.1093/molbev/msw056.