The BaitFisher package manual – Draft II

BaitFisher version 1.2.7
BaitFilter version 1.0.5

March 2016

Christoph Mayer, 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 in MSAs. The main goal of BaitFisher is to avoid redundancy in the construction of baits by designing fewer baits in conserved regions of the MSAs and designing more baits in variable regions. This makes use of the fact that hybrid enrichment baits can differ to some extends from the target region, which they should capture in the enrichment procedure. By specifying the allowed distance between 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 for details.

BaitFilter was designed (i) to determine whether baits bind unspecifically to a reference genome, (ii) to filter baits that only have partial length matches to a reference genome, (iii) to determine the optimal bait region in a MSA and to convert baits to a format that can be uploaded at a bait constructing company. 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.

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., Wägele, 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. doi:10.1093/molbev/msw056

II. Where to obtain the BaitFisher package

The BaitFisher package can be downloaded from
https://github.com/cmayer/BaitFisher-package.git
We recommend to download the ZIP-file:
https://github.com/cmayer/BaitFisher-package/archive/master.zip

III. Compiling and installing BaitFisher and BaitFilter

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 enter the BaitFisher-package-master directory. Now type make on the command line. This compiles the BaitFisher as well as the BaitFilter program. The executables of these to programs care called: BaitFisher-v1.2.7 and BaitFilter-v1.0.5. They can be copied on your computer to any place you like. To use these executables they have to be addressed with its full path, or they have to be placed somewhere where in your system path.

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.

Target MSAs:
MSAs for which baits shall be designed.

Continuous vs. non continuous sequence data:
In most cases it is assumed that you have continuous sequence data, i.e. your sequences are found in exactly the same way in the genomes of the corresponding species. For transcriptomic data it is not guaranteed that the sequences are continuous, since the splicing machinery might have altered the original sequence by removing introns. In this case you might want to cut the alignments with the aid of annotated features (see Section IX) in order to guarantee that baits are constructed for continuous sequence segments. Alternatively one can use BaitFilter to remove bait-binding loci which do not have full-length or at least nearly full-length matches of baits to a reference genome. Both approaches will solve the problem of potentially non-continuous sequences in the MSAs.

Target region:
Genomic DNA region of interest in.

Bait:
An artificially assembled RNA sequence that binds to parts of the target region. It can be used to capture sheared genomic DNA fragments, which hybridize against it.

Tiling design:
A single bait is often not sufficient to capture a target region in unknown genomes. If the researcher knows the sequence of the target region in similar species, he will usually design multiple successive baits to increase the capture effeciancy of DNA fragments even if the sequence has mutational differences to the know sequence. The tiling design specifies how successive baits are arranged (see **Figure** 1) and is completely defined by the following information: bait length, number of successive baits, offset between starting coordinates.

Bait region:

A genomic DNA region for which baits have been successfully determined for a full tiling design. The length of the bait region is identical to the length of the tiling design (see **Figure** 1).

Reference genome:
If alignment cutting is requested, an annotated genome is required. See Section V for details. This genome will be called a reference genome, since its features will serve as a reference for determining feature coordinates in the target MSAs.

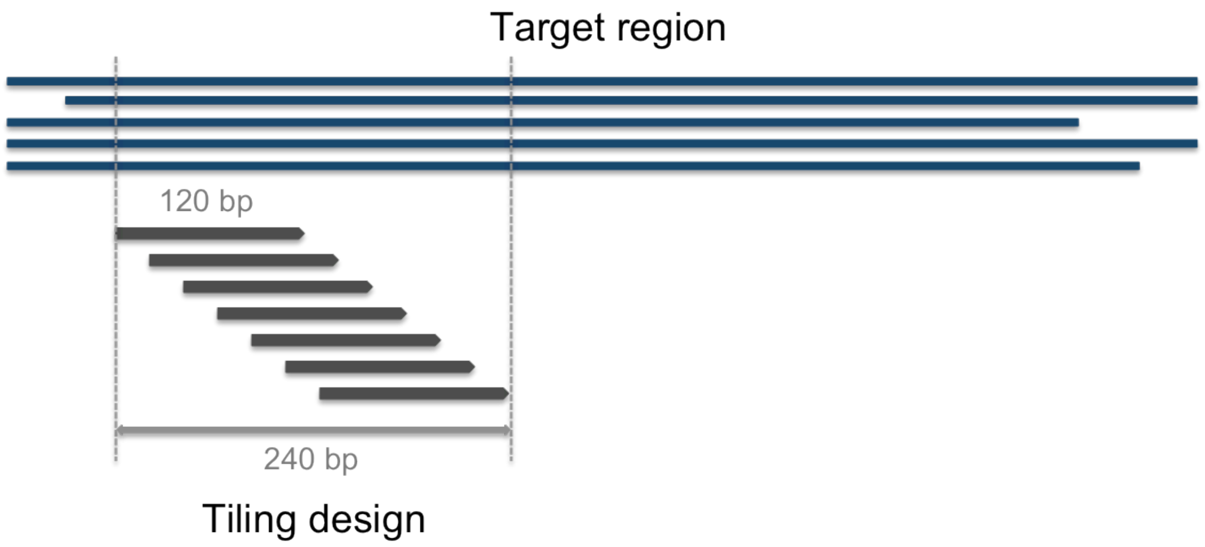

Figure 1: Target region and tiling design. In this example the tiling design consists of seven successive 120-bp baits. The complete tiling design spans 240 bp. Starting coordinates of the baits are separated by an offset of 20 bp. A region for which baits have been constructed according to a given tiling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

V. Special mode – alignment cutting according to features.

There are several reasons why the user might want to cut your alignment according to annotated features prior to bait design. One reason could be that you have transcriptomic MSAs containing non- continuous sequences . In this case one might prefer to design baits only for annotated CDS regions and not for the whole alignment.

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. For each MSA (usually a gene), BaitFisher searches for the corresponding annotation in the gff file.
BaitFisher excised 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 fits to the reference sequence allow to determine the feature coordinates in the MSA. The feature can be excised and bait design can be confined to all features.

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 coded in the sequence name of this reference species. The gene name is required to locate the annotation of this gene in the gff file.It must be possible to split the sequence names of the reference species 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 in exactly 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 “Partent = 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 sequence from the specified genome. This sequence segment will be aligned to the reference sequence in the MSA. The coordinates of where 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.

VI. Special mode – required taxa

BaitFisher can only design baits for sequences that do not contains gaps or ambiguous characters in the bait window. For sequences that do not fulfil this condition BaitFisher will simple not construct baits. If the user considers specific sequences in the MSAs to be of particular importance 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 corresponding MSA or 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 the corresponding 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 one of the two taxa Taxon5 or Taxon6 are present.

VII. 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) (Optional) 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.

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 absolute 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 center 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.

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-deliminator into different parts. One of these parts must be the taxon name, if alignmen tcutting 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 willl divide all sequence names of input files according to the sequence-name-field-deliminator. 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 willl divide all sequence names of input files according to the sequence-name-field-deliminator. 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 minimizes 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 capitalized 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 consits 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 indetical 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.

VIII. 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.

IX. 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.

1st column:	Name of alignment file this bait region belongs to.
2nd column:	Gene name in case of alignment cutting, or the name of the alignment again.
3rd column:	Feature number of requested kind (e.g. CDS) in case of alignment cutting, or 0 otherwise.
4th column:	Maximum number of features of requested kind (e.g. CDS) in case of alignment cutting, or 0 otherwise.
5th column:	Starting coordinate of bait region in the alignment
6th 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.

7th 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.

8th 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 floating point 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.

X. Example analyses

The BaitFisher package includes two example analyses, one with and one without alignment cutting. The example without alignment cutting is included in the BaitFisher package, in the Example/Example-without-alignment-cutting folder. The example with alignment cutting requires a full genome as well as a gff file. They are too large to be included in the BaitFisher download version, so this example analysis has to be downloaded separately from the following link: WILL BE ADDED SOON.

Example without alignment cutting:
In order to run this example anlaysis, unpack the BaitFisher package. On the command line navigate to the BaitFisher-v1.2.7 folder and type make to compile the BaitFisher executable. Then 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 on the command line. If the BaitFisher program has been compiled and not moved out of the BaitFisher-v1.2.7 folder located two hierarchies above the Example-without-alignment-cutting 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 belong to 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 a tiling design. Column 7 shows the number of required baits for this bait region. The minimum number of baits is found for starting position 2. All starting positions have the same sequence coverage in the bait region. 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. BaitFilter and how to proceed after obtaining the BaitFisher output file

After BaitFisher has designed baits for all possible starting positions, we should do some post processing of the output file.

BaitFilter is a command line program that has the following command line options. Options with a description beginning with "(required)" have to be specified in all analyses.

`-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 Bait-Fisher 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: "Agilent-Germany".

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 Bait-Filter 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 Bait-Fisher.

`"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 Bait-Fisher.

`"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.

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

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

`--blast-second-hit-evalue <floating point number>`
Maximum evalue for the second hit. A bait is characterized to bind ambiguously, if we have two good hits. This option is the evalue threshold for the second hit.

`--blast-first-hit-evalue <floating point number>`
Maximum evalue for the first hit. A bait is characterized to bind ambiguously, if we have two good hits. This option is the evalue threshold for the first hit.

`--blast-min-hit-coverage-of-baits-in-tiling-stack <floating`
point number>
Specifies a minimum hit coverage for the primary hit at least one bait has to have in each tiling stack. If one tiling stack of a bait region fails to have a bait with this minimum coverage, the whole bait region is discarded. If not specified, no hit coverage is checked.

Example: 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 conjunction with other filters. 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: ab, as, fb, fs the coverage is checked *before* determining the optimal bait region.

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.

--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 Bait-Filter program. makeblastdb takes the fasta file and creates data base files out of it.

--blast-extra-commandline <string>
When invoking the blast command, extra commandline parameters can be passed to the blast command. As an example with this option it is possible to specifiy the number of threads the blast command should use.

--blast-evalue-cutoff <floating point number>
When invoking the blast command, a default value of twice the blast-first-hit-evalue is used. This should guarantee that all hits necessary for the blast filter are found. However, for the coverage filtering a smaller threshold might be necessary. This can be specified with this option.

-B <string>, --blast-executable <string>
Name of or path+name to the blast executable. Default: nblast. Minimum 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>
In the conversion mode Agilent-Germany each converted file should get
a unique ProdeID prefix, since even among multiple files, ProbeIDs are
not allowed to be identical. This option the user is able to specify a prefix string to all probe IDs in this file.

--verbosity <unsigned integer>
The verbosity option controls the amount of information Bait-Filter 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.

Example applications:
(i) We might want to search for baits that bind unspecifically. 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 unspecifically.
The command used to conduct this analysis is the following:

```
BaitFilter-v1.0.4 -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_Nas 2> e_filter_Nas
```

This command conducts a blast filter run with mode "blast-l", i.e. bait regions that contain one bait with a best hit evalue smaller than 0.00000001 and a second good hit with an evalue smaller than 0.00001 are removed from the bait-file. The blast program 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_Nas and the standard error is redirected to the file e_filter_Nas.

(ii) We might want to ensure that all baits that are used map in full length to a given reference genome. This can be done if we are not sure that the MSA we used for bait design is indeed continuous gDNA. The hit coverage is always check in conjunction with another mode. See comman line option above.

Example:

```
BaitFilter-v1.0.4 --blast-min-hit-of-bait 0.84
  -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_Nas 2> e_filter_Nas
```

In this example the hit coverage is checked after the blast filter we discussed in (i).

(iii) If we still have a bait file in which baits are designed for all possible starting points of bait regions, we likely want to reduce the bait regions to one per feature, gene/alignment. BaitFilter has two optimality criteria: Minimize the number of required baits in the target region or maximize the number of sequences used to construct the baits.

Example:

```
BaitFilter-v1.0.4 -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.

XII. 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 commandline. 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.

XIII. 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. phylogeneic analyses, or does it have to be removed in the way primer sites had 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.