Detailed description of BaitsFinder

- Minor modifications may be necessary depending on the features of the used datasets.
- For the tasks described in section 1 (details may differ strongly among study groups and are here described for the two study groups used by us, Orobanchaceae and Asteraceae), for the used python scripts the name(s) of the input and the output file(s) have to be changed within the script. For the tasks described in sections 2 and 3, names of input files are provided in a config file unless otherwise noted.
- At several steps, BLAST searches are conducted; BLAST is available from http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastNews.
- Names of files and their content may have to follow specific conventions; if so, this is indicated at the respective steps.

Descriptions of the pipeline steps adhere to the following format:

1.1 Description of task

`Command(s)`

Note: optional; further explanations and/or comments on script and/or program usage and/or on input and output files.

Ex.: example(s)

Note: optional; explanations and/or comments on the presented example

1. Identifying single (low) copy genes (SCGs) in species that are closely related to the group of interest and for which whole genome data are available (henceforth called reference species)

Note: This section can be skipped, if reference data (i.e., single copy genes from reference species) are already available.

1.1. Generate a database of SCGs from one or more well-annotated genomes (henceforth called master species, e.g., *Arabidopsis thaliana*):

`formatdb -i input_file -p T`

Ex.: formatdb -i arabidopsis_single_copy_genes.fasta -p T

Note: "arabidopsis_single_copy_genes.fasta" was obtained from http://cgpdb.ucdavis.edu/COS_Arabidopsis/

1.2. Run BLASTP search with the reference species as query:

`blastall -p blastp -d subject –i query -e 1e-10 -o output_file -F "m S" -v 24 -b 24 -g T –m 8`

Note: It is desirable to obtain maximally long alignments of subject and query sequences. Therefore, to prevent breaking long alignments into several smaller alignments due to intervening low-complexity regions soft masking (-F "m S"), which masks low-complexity regions in the word seeding, but not the extension phase, is used. Additionally, gaps are allowed (-g T) to accommodate length differences of genes in more or less distantly related focal and reference species.

Ex.: blastall -p blastp -d arabidopsis_single_copy_genes.fasta –i Mguttatus_256_v2.0.protein.fasta -e 1e-10 -o Mguttatus.arabidopsis -F "m S" -v 24 -b 24 -g T -m 8

1.3. Filter redundant entries of query sequences (i.e., from the reference species) in the output that are due to different translations (i.e., alternative splicing isoforms), retaining only the longest isoform (“primary transcript”) using a custom python script:

`1.3_removeX.p.py`

Ex: "Mguttatus.arabidopsis", obtained in the previous step, as input file, creating "Mguttatus.arabidopsis.1-p" as output file

1.4. Remove query sequences (i.e, from the reference species) with more than one hit with sufficiently high identity scores using a custom python script:

`1.4_id25_0207sort.single.p.py`

Note 1: This is achieved in two steps: in the first step, genes of the reference species with identity to genes from the master species (here *Arabidopsis*) of maximally 25% are removed; in the second step, genes from the reference species with more than one hit to the master species are removed.

Note 2: This script requires two input files: one containing the cleaned query sequences generated in step 1.3, the second the protein sequences of the reference species. This script creates two output files, one containing a list of proteins from the reference species that have only a single significant hit to a gene from the master species, the second containing the IDs of these sequences.

Ex.: "Mguttatus.arabidopsis.1-p ", obtained in the previous step, and "Mguttatus_256_v2.0.protein.fasta" (downloaded previously from Phytozome) as input files, creating "Mguttatus.arabidopsis.single.gene.list.fa" and "Mguttatus.arabidopsis.single.gene.list" as output files.

1.5. Remove putatively duplicated genes from the reference species using BLAST and a custom python script.

```
formatdb -i list_of_putative_SCGs_created_in_previous_step -p T

blastall -p blastp -d list_of_putative_SCGs_created_in_previous_step
-i query -e 1e-10 -o output_file -F "m S" -v 24 -b 24 -g T –m 8

1.5_id25_0207sort.single.p-2.py
```

Note 1: In a first step, reference sequences are blasted against each other. In a second step, query sequences with more than one hit with sufficiently high identity are removed. The thus created list contains the single copy genes from the reference species.

Note 2: This script requires three input files: the first one contains the results from the previous BLAST, the second contains the filtered list of genes from the reference species obtained in step 1.3, and the third contains the protein sequences of the reference species. The script creates two output files, one containing a list of SCGs from the reference species, the second containing the IDs of these sequences. The script uses the same two step procedure as described for step 1.4.

Ex.: formatdb -i Mguttatus.arabidopsis.single.gene.list.fa -p T

blastall -p blastp -d Mguttatus.arabidopsis.single.gene.list.fa -i Mguttatus.arabidopsis.single.gene.list.fa -e 1e-10 -o Mguttatus.vs.Mguttatus -F "m S" -v 24 -b 24 -g T –m 8

"Mguttatus.vs.Mguttatus" (i.e., the list of SCGs from the reference species), "Mguttatus.arabidopsis.1-p" (i.e., the list of genes from the reference species after removal of alternative splice isoforms), and "Mguttatus_256_v2.0.protein.fasta" (i.e., the protein sequences of the reference species downloaded from Phytozome) as input files, creating "Mguttatus.arabidopsis.single.gene.list.-2.fa" and "Mguttatus.arabidopsis.single.gene.list.-2" as output files.

OPTIONAL: If additional reference species are to be included, a joint list of single copy genes can be created, where homologous genes from different reference species are present only once. The following steps are needed:

1.6. Obtain a list of single copy genes from additional reference species following steps 1.1 to 1.5.

Note: Step 1.3 is not necessary, if there are no alternative splicing isoforms indicated.

Ex.: Applying steps 1.1, 1.2, 1.4 and 1.5 to *Solanum lycopersicum* (using "ITAG2.4_proteins.fasta" downloaded from the Sol Genomics Network: ftp://ftp.solgenomics.net/genomes/Solanum_lycopersicum/annotation/ITAG2.4_release/) results in "ITAG.arabidopsis.single.gene.-2.fa", a list of single copy genes in *S. lycopersicum*, and "ITAG.arabidopsis.single.gene.list.-2", containing the IDs of these sequences.

1.7. Identify unique single copy genes (i.e., those not already present in the primary reference species) using BLAST:

```
formatdb -i single_copy_genes_primary_reference_species -p T
```

```
blastall -p blastp -d single_copy_genes_primary_reference_species –i
secondary_reference_species -e 1e-10 -o output_file -F "m S" -v 24 -b
24 -g T -m 8
```

Note: The list of single copy genes from the primary reference species (here: *Erythranthe guttata*) has been obtained in step 1.5.

Ex.: formatdb -i Mguttatus.arabidopsis.single.gene.list.-2.fa -p T

blastall -p blastp -d Mguttatus.arabidopsis.single.gene.list.-2.fa –i ITAG.arabidopsis.single.gene.list.-2.fa -e 1e-10 -o ITAG.-2.vs.Mguttatus.-2 -F "m S" -v 24 -b 24 -g T -m 8

1.8. Remove SCGs from the secondary reference with high similarity to SCGs from the primary reference using a custom python script

```
1.8_removeS.py
```

Note: This script requires two input files: one containing the list of sequences from the second reference having significant hits with sequences from the primary reference (i.e., the blast output file generated in step 1.7), and the other containing the IDs of single copy sequences from the second reference species generated in step 1.6. The script produces as output file a list of IDs of genes from the second reference species that have no hit to genes from the primary reference species.

Ex.: Using "ITAG.-2.vs.Mguttatus.-2", created in step 1.7, and "ITAG.arabidopsis.single.gene.list.-2", generated in step 1.6, to generate the output file "ITAG_nohit".

1.9. Extract protein sequences of the secondary reference species using a python script.

```
1.9_extract proteins.py
```

Note: This script requires two input files: one containing the list of IDs of genes from the second reference species that have no hit to genes from the primary reference species, generated in step 1.8, the second containing the protein sequences of the secondary reference species (already downloaded from Phytozome). This script produces a single output file, containing title and protein sequences of the unique SCGs from the secondary reference species.

Ex.: "ITAG_nohit" and "ITAG2.4_proteins.fasta" as input file, creating "ITAG_sorted" as output file.

1.10. Merge lists of single copy genes from the primary and the secondary references species.

Ex.: "Mguttatus.arabidopsis.single.gene.-2.fa" and "ITAG_sorted" were merged to "Mguttatus_Solycdual.sorted.fa" (corresponding to "reference. protein" in the example data provided with the scripts) using, e.g., "cat Mguttatus.arabidopsis.single.gene.-2.fa ITAG_sorted > Mguttatus_Solycdual.sorted.fa".

1.11. Merge lists of coding sequences (CDS) from the primary and the secondary references species.

Ex.: "Mguttatus_256_v2.0.cds.fasta" (downloaded from Phytozome) and "ITAG2.4_cds.fasta" (downloaded from the Sol Genomics Network) were merged to create "Mguttatus_Solycdual.sorted.cds" (corresponding to "reference.cds" in the example data provided with the scripts).

Note: The following steps are conducted by two custom python scripts (`blast_tcl.py` for step 2.1 and `combined-ini.py` for steps 2.3–7.1) that use the same config file (config.ini): more detailed information on the arguments provided in the config file can be found in the example config file distributed with the scripts. The two python scripts, the config file, the program tcl_blast_parser_123_V047.tcl (available from http://code.google.com/p/atgc-tools/downloads/detail?name=tcl_blast_parser_123_V047.tcl) and an input folder (whose name is defined in the config file) containing all reference and query data have to be located within the same folder. The example sections use the files from the example data provided with the scripts (one reference species, "reference", and four focal species, the non-parasitic "U1S1" and the parasitic "U2S2", "U3S3", and "U4S4").

2. Identifying single copy genes in focal species.

Note 1: File names of the focal species (in our study: four Orobanchaceae or three Asteraceae species) should adhere to the following naming conventions: (1) the first four positions of the file name have to be unique among the set of included focal species; (2) the sequence IDs within these files should be of the same form as in the file name.

> Ex.: The files "GB_ESTs_Feb_2007.sp.Cart_tinc.clean.assembly" and "GB_ESTs_Feb_2007.sp.Heli_annu.clean.assembly" were renamed to "Cart_tinc.clean.assembly" and "Heli_annu.clean.assembly" to ensure unique file name beginnings. Within "Cart_tinc.clean.assembly", unigene IDs were changed to all start with "Cart_tinc" (e.g., "CART_TINC" was changed to "Cart_tinc").

Note 2: Irrespective of the origin of the reference data, IDs in the protein and cds files should adhere to the structure "string1"."string2", while the IDs of the gff3 file should adhere to the structure "string1"."string2"."TAIR".

> Ex.: The IDs in the reference protein file were changed from "Migut.D00685.1.p" to "A.D00685" and from "Solyc12g099490.1.1" to "A12g099490.1", those in the cds file from "Migut.D00685.1" to "A.D00685" and from "Solyc12g099490.1.1" to "A12g099490.1", those in the gff3 file to "A.D00685.TAIR" and "A12g099490.1.TAIR".

2.1. Run BLASTX search of unigenes from each of the focal taxa separately against the reference and filter out data with insufficient quality using a custom python script:

`blast_tcl.py`

> *Note 1*: This script uses input as defined by arguments 1, 4, 6, 7, 10 in the config file. Except for input and output file names, parameters for BLAST (except for gapped versus ungapped blast, which is taken from the config file) and for the blast parser[1] are hard-coded in the script and have to be changed there, where needed.

---

[1] If the blast parser is run outside this python script, the stem for the output file names has to end with "gap.out" or "ungap.out", depending on the used blast strategy.

Ex: tclsh tcl_blast_parser_123_V047.tcl U1S1_ungap.out U1S1_ungap.out 20 40 100 MATRIX

Note: The matrix file " U1S1_ungap.out.matrix.identity" contains primary hits if they satisfy each of the following conditions: identity at least 40, expectation better than 1e-20 (the cut-off is given as –log(e-value), i.e., 20), and an alignment length of at least 100 positions (at the amino acid level); the BLAST file

Note 2: Whether to use ungapped or gapped BLAST (argument 6 in the config file) will depend on the study design. As outlined under step 1.2, gapped BLAST will allow for longer alignments and thus result in more data, while ungapped BLAST may be preferable if more conserved baits and fewer data are needed.

Note 3: The blast parser produces several output files (for details see http://www.atgc.org/BlastParser/Blast_Parser_017.html), of which only the *matrix* file (contains information on primary hits only organized in five columns with query ID, subject ID, identity (normalized between 0 and 1), normalized expectation, and alignment length, respectively) and the *all_hits* file (contains information on all hits organized in 14 columns, including those pertaining to start and end points of the alignments of query and subject) are needed for the purpose of this pipeline.

Ex: Running this script using an input folder containing "U1S1.fasta", "U2S2.fasta", "U3S3.fasta", U4S4.fasta", and "reference.protein" as input files creates in a separate output folder ("blast_tcl_out") the output files e.g." U1S1_ungap.out" for ungapped blast; "U1S1_gap.out" for gapped blast and numerous blast parser files.

2.2. Blast unigenes of focal species against data from suspected contaminating species (e.g., host species in case of parasite focal species) to identify putative contaminant sequences. This step is optional.

```
formatdb -i contaminant_species -p F
```

```
blastall -p blastn -d contaminant_species -i focal_species -e 1e-100
-o output_file -v 24 -b 24 -g T
```

Note 1: Using gapped BLAST (-g T) makes this approach more conservative by allowing more deviation from reference contaminant species.

Note 2: The names of the blast output files should adhere to the following naming convention: query name + ".sort_" + the first two letters of the file name of the contaminating species + ".out" (e.g., "U3S3.sort_H2.out").

Ex.: formatdb -i Sbicolor_255_v2.1.cds.fa -p F

blastall -p blastn -d Sbicolor_255_v2.1.cds.fa -i StHeBC2.fasta -e 1e-100 -o StHeBC2.sort_Sb.out -v 24 -b 24 -g T

Note 3: If more than one contaminant species is to be tested, their data should be merged in a single file beforehand.

---

processing ended at the matrix level (argument "MATRIX") to avoid potentially time-consuming and for our purpose unnecessary sequence clustering. The second output file of interest is "U1S1_ungap.out.all_hits".

2.3.–7.1. The following steps (from removing putative contaminant sequences, if needed, to bait sequence generation) are achieved within a single custom python script. Files and folders created during intermediate steps are saved in a temporary folder (defined in the config file in argument 3), the final bait sequences are saved in an output folder (defined in the config file in argument 2).

`combined-ini.py`

> *Note*: This script uses input as defined by arguments 1–3, 5–16 in the config file (see example config file provided with this script for the used arguments). The location of some files is not indicated in the config file, but these have to be present in input folder; these files are: the *all_hits* files and *matrix* files generated in 2.1 as well as the putative contaminant sequences generated in 2.2 (e.g., U1S1_ungap.out.all_hits and U1S1_ungap.out.matrix.identity for ungapped blast; U1S1_gap.out.all_hits and U1S1_gap.out.matrix.identity for gapped blast; U2S2.sort_H1.out).

2.3. Combine the *all_hits* files of each focal taxon into a single file.

> *Note:* This step requires the *all_hits* files as input and generates in the temporary folder two subfolders: "clean", containing the *all_hits* file with unigenes without any hit to the reference being removed; "combine", where the cleaned *all_hits* files have been combined to a single file.

2.4. Remove putative contamination from contaminant species (e.g., host species).

> *Note*: This step is necessary for parasitic species, but may also be used for non-parasitic species if contamination from other sources may be an issue. If no contaminant species is indicated in the config file (i.e., argument 11 is set to "Plant_ver=2" and in argument 12 the name of the contaminant species data is set to "NAN" [e.g., "U1S1.fasta=host_name=NAN"), output files from the focal species will be created named as if contamination (e.g., from host species) has been removed (see step 2.4.3; steps 2.4.1 and 2.4.2 are entirely skipped).

2.4.1. Construct a joint list of putative host sequences.

> *Note*: Based on the blast output files of focal species against contaminant species generated in step 2.2, sequences from focal species with identity to sequences from contaminant species of at least 95% (the identity cut-off used is set in the config file in argument 15 and can be changed as needed) are regarded as contaminations. Output files contain a list of IDs of sequences from the focal species that are assumed to be contamination from a host.
>
> > *Ex*: This step generates output files for the three parasite species U2S2, U3S3 and U4S4 ("U2S2.sort_H1.out.sort", "U3S3.sort_H2.out.sort" and "U4S4.sort_H3.out.sort").

2.4.2. Merge host sequences into a single file.

> *Note*: This step combines the IDs from sequences of the focal species that are assumed to be host sequences generated in 2.4.1. into a single file.
>
> > *Ex*: This step generates output file "all_host".

2.4.3. Remove host sequences.

> *Note*: This step generates files from focal species, where putative contaminant sequences have been removed.
>
> > *Ex*: This step generates modified *matrix* files "u2_ungap_host_free"

2.5. Identify and remove putative paralogues.

Note: Different unigenes from the same focal taxon that have been blasted against the same reference single copy gene with a user-defined overlap cut-off are considered as putative paralogues.

2.5.1. Add position information to *matrix* files.

Note 1: The *matrix* files generated in steps 2.1. or 2.4.3., respectively, do not contain information on the alignment positions, which is, however, available in the *all_hits* files. An output file corresponds to a *matrix* file with four additional columns pertaining to alignment start and end points of focal and reference sequences.

Ex: This step generates modified *matrix* files "u1_ungap_host_free.add".

2.5.2. Identify overlapping query sequences that blast against the same reference gene (thus indicating putative paralogy).

Note 1: The extent of permitted overlap determines how strictly the paralogy filter is applied. We use here a permitted overlap of maximally 3 bp (this number can be changed in the config file in argument 14).

Note 2: In a first step, entries of the input file (containing data from one focal species only) are sorted in ascending order using firstly the gene-ID of the reference species (i.e., the subject ID, in the fourth column) and secondly the alignment start point of the subject sequence (i.e., from the reference species; in the fifth column). In a second step, for each gene of the reference species it is checked whether the alignment end point in row *n* is larger by at least 3 than the alignment start point in row *n*+1: if yes, then the sequences from the focal species in rows *n* and *n*+1 are considered putative paralogues, which is indicated with an entry in the last column of the created output file.

Ex: The first step generates "u1_ungap_host_free.add.sorted", the second one generates "u1_ungap_host_free.add.sorted.paralogInfo".

2.5.3. Remove putative paralogues.

Note: If any of the query sequences (i.e., unigenes) sharing the same reference gene (i.e., subject) is indicated as putative paralogue, the entire set of unigenes aligned to the same reference gene is removed from the focal species.

Ex: This step generates "u1_ungap_host_free.add.sorted.paralogInfo.out".

2.5.4. Combine the paralogue-free data generated in 2.5.3 into a single file.

Ex: This step generates "4_nogap_hostfree_3sorted".

3. Extract sequences from both reference and focal species

3.1. Extract sequences of the non-paralogues genes from each of the focal species.

Note: This step generates output files containing only the aligned part of the sequences from the focal species.

Ex: This step generates "U1S1_nogap_edit.codons.query_all.seq".

3.2. Combine, per gene, sequences of the reference species and of all focal species into a single file (i.e., one file per locus).

Note: This step creates a temporary output folder named "group1" containing one fasta file per gene. Each fasta file contains the newly extracted gene sequence from the reference species (start and end point are the minimum start and maximum end point over all alignments to sequences from the focal species, i.e., this sequence from the reference species may contain stretches without any aligned positions from the focal species) and all aligned sequences from the focal species.

3.3. Bring sequences to the same orientation relative to the reference species.

Note: As some (or all) of the sequences from the focal species may be aligned to the reference species as reverse complement, re-orientation is necessary prior to the alignment to be conducted in step 4. This script goes through the fasta files created in step 3.2 and produces reverse complements for those sequences of the focal species, whose BLAST alignment endpoint on the reference species is smaller than the start point. The thus altered fasta files are written into a temporary folder "group2".

OPTIONAL: If data quality from some of the focal species is non-satisfactory (e.g., too many short unigenes), data from genes that (apart from the always present reference species) only contain such focal species may be considered insufficiently reliable and may be removed. If this part is needed, a single highest quality focal species, which is used to retain only those genes where (in addition to the reference species) data from at least this focal species are present, has to be defined.

3.4. Retain only genes where data from focal species with sufficient data quality are present.

Note: The single best-quality focal species is defined in the config file (argument 13); if this step is to be skipped, instead of a file name the argument "NAN" is given. This step generates a temporary folder "group3" containing only fasta files for those genes that contain data from suitable focal species; in case no best-quality focal species is defined, the folder "group3" has the identical content to folder "group2".

4. Sequence alignment

Note: When identifying single copy genes from the focal species, in step 2.1 either ungapped or gapped BLAST can be used. In case of ungapped BLAST, sequences from reference and focal species can be aligned unambiguously, as there are no length differences between aligned segments; consequently, also sequences from focal species can be aligned unambiguously, although sequences from different focal species may only partly overlap or even not overlap at all, as they align to different sections of the sequence from the reference species. For gapped BLAST, putatively present length differences between aligned sequences of reference and focal species may cause ambiguities in the alignment not only between reference and focal species, but especially among focal species. Therefore, a dedicated alignment program is used to re-align all sequences. Accordingly, different steps are required, which will be described in turn ("u" indicating steps necessary in case of ungapped BLAST, "g" indicating steps necessary in case of gapped BLAST).

4.u1. Align sequences of the focal species and the reference species using information on start and end points of the BLAST alignment.

> *Note 1*: This step does two things: First, it aligns sequences from focal and reference species based on the BLAST alignment by adding gaps at the beginning and/or the end of the shorter sequence; second, white spaces in the sequence titles are replaced with asterisks (necessary for later steps), i.e., "Subject_ID Subject_start_site-Subject_end_site length_of_Subject" becomes "Subject_ID*Subject_start_site-Subject_end_site*length_of_Subject" (e.g., ">U1S1_97231 546-1703 forward [A.A00009*232-1389] length=1158" becomes ">U1S1_97231*546-1703*forward*[A.A00009*232-1389]*length=1158"). This step creates "group5" as temporary output folder.

4.g1. Align sequences of the focal species and the reference species using MAFFT.

> *Note 1*: The path to the MAFFT executable is provided in the config file (argument 5). For alignment, we have chosen as iterative refinement method E-INS-I (i.e., options --ep 0 --genafpair --maxiterate 1000), which is suitable for data sets containing multiple conserved domains and long gaps.
>
> *Note 2*: The step creates a temporary output folder "mafft_out" containing fasta files of aligned sequences.

4.g2. Change the MAFFT output from multi-line to single-line fasta.

> *Note*: This step creates a temporary output folder "group4" containing the modified fasta files.

4.g3. Changing white spaces in the sequence titles to asterisks (necessary for later steps).

> *Note 1*: The step creates a temporary output folder "group5", containing fasta files of aligned sequences with modified sequence names.

5. Extract exon sequences and remove those of insufficient length

Note 1: Exon sequences are extracted based on information on exon limits available in the gff3-files of the reference species (a combined file in case of more than one reference species).

> Ex: For the Orobanchaceae part, "Mguttatus_256_v2.0.gene.gff3" and "Slycopersicum_225_iTAGv2.3.gene.gff3" have been combined into a single file "reference.gff3", for the Astercaeae part "Athaliana_167_TAIR10.gene.gff3" has been used (all downloaded from Phytozome10.3: https://phytozome.jgi.doe.gov/pz/portal.html).

Note 2: IDs of the gff3 file should adhere to the structure "string1"."string2"."TAIR".

> Ex.: The IDs in the reference protein file were changed from "Migut.D00685.1.p" to "A.D00685" and from "Solyc12g099490.1.1" to "A12g099490.1", those in the gff3 file to "A.D00685.TAIR" and "A12g099490.1.TAIR".

5.1. Extend sequences to start with the first position of the cds.

> *Note 1*: Sequences from the aligned reference species may not start with the first position of the entire coding region (cds). To simplify subsequent steps, the sequences from the reference and the focal species are extended at the beginning by inserting plus signs making alignment position 1 corresponding to cds position 1.
>
> > Ex.: >A.A00009 starts at cds position 232, accordingly 231 "+" have to be inserted at the beginning of this sequence, from the reference species, and those from the focal species aligned to it.
>
> *Note 2*: This step creates a temporary output-folder "filling-in" containing the extended aligned sequences.

5.2. Transpose the aligned sequences.

> *Note 1:* To make subsequent steps easier, the sequence alignments are transposed. The resulting file has as many columns as aligned sequences, its first row contains the sequence names and rows 2 to n+1 contain the first to n^{th} alignment position.
>
> *Note 2*: This step creates a temporary output-folder "group6" containing the transposed alignments.

5.3. Clean the files with transposed sequences.

> *Note*: This step creates a temporary output-folder "group7" containing the transposed alignments, where empty lines, if present in the input, have been removed.

5.4. Compile start and end points of all exons per gene in a single file.

> *Note*: This step uses gff3-files from the reference species as input files and creates a single output file "map" containing gene name, gene orientation, and start and end points of the gene's exons (one line per gene).

5.5. Add flags for split site positions in the transposed alignments.

Note: This step creates a temporary output folder "group8" containing the transposed alignments, where additionally the end-points of each exon are indicated by an additional line containing an "S" for each sequence.

5.6. Extract exons into separate files.

Note: This step creates a temporary output-folder "group9" containing separate transposed alignment files from each exon.

Ex.: File "A.B00613", pertaining to a gene with 3 exons, will be separated into "A. B00613.1", "A. B00613.2" and "A. B00613.3".

5.7. Remove exons of insufficient length.

Note 1: The desired bait length depends on the study design and may vary from 80 to 120 bp; the length threshold below which an exon is removed should be adjusted accordingly. This value (here, we use 120 bp as cut-off) can be changed in the config file (argument 16).

Note 2: This step creates a temporay output-folder "group10" containing only those exon alignments that exceed a certain threshold.

Ex.: If lengths of the three exons "A.A00009.1", "A.A00009.2" and "A.A00009.3" are 210 bp, 105 bp and 300 bp, only "A.A00009.1" and "A.A00009.3" will be retained.

6. Extract baits

6.1. Re-transpose files containing the exon sequences to fasta files.

Note 1: Alignments have been transposed in step 5.2 to make extraction of exons easier. From here on, the transposed orientation is no longer needed and is, therefore, reverted.

Note 2: This step creates a temporary output-folder "group11" containing only those exon alignments in standard fasta format.

6.2. Remove gaps in exons:

Note 1: In order to obtain baits that might be targeting different focal species, gaps in the exons have to be removed.

Note 2: This step creates a temporary output-folder "group12" containing the alignments of exons without gaps, and "group13" containing the alignments of sufficiently long exons without gaps (here 120 bp, this can be changed in the config file in argument 16).

6.3. Extract baits.

Note 1: Tiling density (here 2×, i.e., a 60 bp overlap) depends on the study design. To avoid redundancy between baits, a threshold is set for the maximally permitted overlap of baits (here 80 bp, i.e., from a fragment of 160 bp still two baits, overlapping by 80 bp, could be extracted); this threshold is hard-coded in the script (in section 6.3) and has to be changed there, if needed.

Note 2: This step creates a temporary output-folder "group14" containing the baits from one exon of one focal species (e.g., a gene with five exons and each exon with data from three focal species will result in 15 output files). In the second step, all baits of the focal species are combined into a single fasta file "nogap_baits.combined.fasta", in the final folder.

6.4. Combine exon sequences into single files per species.

Note: This step combines all exons of a single focal species into one file. It takes the folder containing the alignments of sufficiently long exons without gaps created in step 6.2 as input and creates fasta files per species containing all exons.

Ex: This steps creates "U1S1.fasta", "U2S2.fasta", "U3S3.fasta", "U4S4.fasta".

7. Data cleansing

7.1. Remove genes with too few baits in a best-quality focal species.

Note 1: Generally, too short loci that should not be targeted have to be removed, especially if these are an artefact of insufficient data quality. In these cases, only a single focal species, which has the best quality data (i.e., the species defined in the config file in argument 13; see also step 3.4), should be considered for identifying too short loci. In consequence, a gene is removed, if it is too short in this best quality focal species irrespective of the gene's length in any of the other focal species.

The threshold, below which a locus is considered too short, will depend on the study design. Here we use the number of baits per species as criterion: although the number of baits will also depend on exon structure (i.e., a gene with many short exons will yield fewer baits than a gene with a single long exon, even if the total length of the cds is the same), the number of baits will correlate with gene length. Here, we use a cut-off value of four baits (corresponding to at least 280 bp per gene, i.e., the minimum length if the baits are from a single exon and the last bait has an overlap of 80 bp with the preceding bait); this cut-off value is hard-coded in the script (in section 7.1) and has to be changed there, if needed.

Note 2: This step takes the fasta file containing the baits per species created in step 6.3 as input and creates an output file in the final folder "nogap_sorted.result.fa" containing only baits for those genes that have at least four baits in the best- quality focal species. If no single best-quality focal species is defined, this step will be skipped and "nogap_sorted.result.fa" will not be created.

7.2. Remove redundant baits (i.e., having a similarity above a certain threshold) using CD-HIT-EST.

Note: We used online CD-HIT-EST (available at: http://weizhong-lab.ucsd.edu/cdhit_suite/cgi-bin/index.cgi?cmd=cd-hit-est). As baits from different genes, but with similar sequences, may cause cross-hybridization during enrichment, only one of those should be retained. The precise value of the cut-off depends on study design; here, we used ≥90% (minIdentity=90; other settings were left at their default).

Ex.: "nogap_sorted.result.fa" obtained in previous step as input file, generating "1456519283nogap.fas.1" as output file containing baits that share less than 90% identities.

7.3. Identify baits with high similarity to plastid sequences using blast.

```
formatdb -i plastid_genomes -p F
```

```
blastall -p blastn -d plastid_genomes –i baits -e 1e-10 -o output  -v 24 -b 24 -g T -m 8
```

Note: It is very important to remove baits that are similar to plastid sequences, because these are present in high copy numbers and will be highly enriched during the enrichment phase, reducing the yield of sequences from single copy nuclear genes. The precise similarity threshold used will depend on the study design.

Ex.: formatdb -i cpgenome.fasta -p F

blastall -p blastn -d cpgenome.fasta –i 1456519283nogap.fas.1 -e 1e-10 -o 1456519283nogap.fas.1.out -v 24 -b 24 -g T -m 8

Note: For the Orobanchaceae study, we used 12 plastid genomes: *Boulardia latisquama, Cistanche deserticola, Conopholis americana, Epifagus virginiana, Lindenbergia philippensis, Orobanche crenata, Orobanche gracilis, Phelipanche purpurea, Phelipanche ramosa, Schwalbea americana, Striga hermonthica, Triphysaria versicolor.*

7.4. Remove putative plastid sequence using a custom python script:

`remove_plastid.py`

Note: This script uses two input files, one containing the sorted baits that share less than 90% identities generated in step 7.2 and another one containing the blast output file of baits against plastid genomes generated in step 7.3.This script generates two output files, one containing the list of genes, which have at least 90% identity with any one of the 12 plastid genomes, another containing baits where sequences of putative plastid origin have been removed.

Ex.: Using "1456519283nogap.fas.1" and "1456519283nogap.fas.1.out" as input to create "1456519283nogap.fas.list" and "1456519283nogap.fas.1.plastid_free" as output files.