Scripts written for and used in the paper

"Nano-Strainer: a workflow for identification of single-copy nuclear loci   
for plant systematic studies, using target capture kits and Oxford Nanopore long reads"  
published by Agnes Scheunert et al. in "Ecology and Evolution"

All scripts written by Ulrich Lautenschlager or Tankred Ott

General information: shell scripts are self-executables and can be called from a Linux command line; R scripts can be executed by typing "Rscript [scriptname].r" in the command line, provided R/Rscript has been added to the $PATH. Output files will normally be generated in the directory where the script is located, unless output paths are given.

Note: supplementary material of the paper as referred to in this help file is either available at Dryad (see Data Accessibility Statement in the paper) or in this repository.

1\_extract\_files.sh

Purpose: extract clusters belonging to pre-choice loci, from collection of all clusters OR

extract FASTA files of candidate loci, from collection of pre-choice loci

Chapter: 2.3.3. / 2.3.4.

Input: folder containing all aligned, renamed cluster files, singleton clusters removed  
executed once for each individual OR

folder containing locuswise concatenated consensus sequences of pre-choice loci ("*prechoiceloci\_clusterconsensuses\_locuswise*") PLUS  
a CSV file listing the names of all elements to be extracted

Execute: ./1\_extract\_files.sh [name\_of\_csv\_file].csv [input\_folder] [output\_folder]

2\_index1\_standardized\_mean\_reads\_per\_cluster.r

Purpose: calculation of index 1, the mean standardized number of reads per VSEARCH cluster in a locus (including low-coverage and singleton clusters). Calculated based on the locus summary statistics of the pre-choice loci (see Supplementary Materials of the study as available at Dryad, file "6\_full\_locus\_statistics.xlsx").

Chapter: 2.3.4.

Input: CSV file giving the total number of reads and the total number of clusters for each individual and locus, taken from the locus summary statistics file; example see file "*2\_input\_example.csv*" (note that this file includes values for all enriched, not only the pre-choice, loci)

Execute: see script; the output can then be added to the locus summary statistics file

3\_index2\_KBS.r

Purpose: calculation of index 2, the k-mer based similarity index

Chapter: 2.3.4.

Input: folder with FASTA files of locuswise concatenated cluster consensus sequences (see Supplementary Materials at Dryad, file "4\_workflow\_commands.docx", folder there named "*prechoiceloci\_clusterconsensuses\_locuswise*")

Execute: see script; the output can then be added to the locus summary statistics file

Dependencies: *ape* and *kmer* libraries in R

4\_index3entropy\_index4silhouette.r

Purpose: calculation of index 3 (entropy) and index 4 (silhouette index)

Chapter: 2.3.4.

Input: same as for script "3\_index2\_KBS.r "

Execute: for single execution with only one FASTA file:

./4\_index3entropy\_index4silhouette.r -k [desired k-mer length, default=8] -l [desired linkage criterion, default=single] -o [output\_prefix] [input\_file]

parameters can be set upon calling the script; for recommendations on choosing *k*-mer lengths and linkage criteria, please refer to the main text of the paper and the detailed methods document "1\_detailed\_methods.docx" at Dryad. The script will also produce output on the screen; the final results can then be added to the locus summary statistics file  
See "4\_workflow\_commands.docx" for usage within a FOR loop

Dependencies: *ape*, *kmer*, *argparse* and *cluster* libraries in R

5\_best\_loci.r

Purpose: score loci based on values from three specialized indices (1-3) and report loci by decreasing order. Note that percentiles are calculated excluding the respective value (">" and "<" instead of "≥" and "≤") by the script.

Chapter: 2.3.4.

Input: Excel file containing index values for each pre-choice locus; example see file "*5\_input\_example.xlsx*"; the columns have to be named "*locus*", "*norm.mean*" (index 1), "*KBS\_mean*" (index 2, mean), "*KBS\_sd*" (index 2, standard deviation) and "*entropy*" (index 3)

Execute: see script; the chosen candidate loci and their statistics can then be added to the locus summary statistics file (sheet "*candidate loci*")

Dependencies: *readxl* library in R

6\_calculate\_dendrograms.r

Purpose: calculate a dendrogram from the cluster consensus sequences of a given FASTA file; iterate over all FASTA files contained in a folder

Chapter: 2.3.5.

Input: folder containing FASTA files of candidate loci (here named "*candidate\_loci*")

Execute: see script; the output TRE files will be generated in the same folder

Dependencies: *ape*, *kmer* and *tools* libraries in R

7\_vsearch\_testclusteringthresholds.py

Purpose: execute a series of VSEARCH clusterings (in "*--cluster-fast*" mode) on a single input file or a folder with input files using different, pre-defined clustering thresholds (CTs)

Chapter: 2.4.4.

Input: a single FASTA file containing sequences to be clustered, or a folder with a collection of FASTA files to be processed sequentially

Execute: for execution in single mode (input one FASTA file only):

7\_vsearch\_testclusteringthresholds.py simple /[path/to/inputfile].fasta /[path/to/outputfolder]/ [--ct 0.50 --ct 0.75] -t [number of threads]

Note: CTs that should be tested must be given one after the other, separated by spaces, here exemplified by CTs = 0.50 and 0.75.  
For use in multi mode see script; the script will generate one folder for each input file and one subfolder for each tested clustering threshold. Clustering is performed with the following settings (which can be modified if desired by altering the VSEARCH\_CALL line in the script): cluster identifier information added to the output files; clusters sorted by decreasing abundance; no masking performed; checking the plus and minus strand during comparison of sequences with the cluster seed.  
The output (for each CT) will contain: files for all generated clusters with sequential numbering, two files containing cluster centroids and cluster consensus sequences, one file containing the aligned sequences of each cluster, clustering results each in uclust-like and BLAST "*outfmt 6*" format, and a LOG file. Results (cluster files) can then manually be examined for sequences separated below a given CT  
Dependencies: Python

8\_filter\_entirelyunmappedreads.sh

Purpose: extract amplicon reads that did not map to any of the references, based on the mapping results of the previous step

Chapter: 2.4.4.

Input: folders from the original mapping procedure, named "*reads*" (contains one FASTQ file per sample) and "*results*" (contains SAM files resulting from the mapping)

Execute: see script; the output, per individual, is one TXT file listing unmapped read IDs (from which the proportion of unmapped reads can be determined), one intermediate TXT file ("*regex.txt*") and, in a folder "*reads\_unmapped*", one FASTQ file with the extracted unmapped reads from the list. After converting these into FASTA files, they can be used for BLASTing against the mapping references, see file "4\_workflow\_commands.docx"

9\_extract\_mapping\_statistics.sh

Purpose: see "4\_workflow\_commands.docx"; from a collection of mapping statistics obtained by executing Samtools flagstat on several mapping files (SAM), extract and summarize the information contained in the single files

Chapter: 2.4.4.

Input: a folder containing mapping statistics files as obtained by running Samtools flagstat

Execute: see script; the output is saved in the directory where the script is located, and is a CSV file named "*mapping\_statistics.csv*" containing, in four tab-delimited columns, the respective name of the flagstat results file and the number of primary mapped, secondary and supplementary mapped reads

Dependencies: the script expects input files as generated by Samtools flagstat v. 1.13

10\_extract\_canu\_statistics.sh

Purpose: extract relevant statistics from a collection of Canu results files

Chapter: 2.4.5.

Input: a folder containing a collection of subfolders (one for each amplified marker), each with again subfolders for each individual, containing the output of the respective Canu assembly run

Execute: see script; the output is one tab-delimited CSV file per marker, each containing, for each sample, 1) the number and 2) percentage of reads which did not pass the *minReadLength* filter, 3) the number of reads loaded into the sequence store ("*SeqStore*"), 4) the number of reads available for unitigging, 5) the number of reads that remained unassembled, 6) the amount of contigs that was assembled, and 7) the number of reads supporting each assembled contig.   
This information can be used to calculate further important statistics (see Supplementary Materials of the study as available at Dryad, file "8\_mapping\_and\_canu\_stats.xlsx")  
The script extracts this information from the *.seqStore.err*, *.report*, *.unassembled.fasta* and *.contigs.fasta* files in each Canu output folder  
Dependencies: the script expects file names and folder structures as in Canu v. 2.1 outputs.

11\_sequencename\_from\_filename.sh

Purpose: replace contig sequence names within a collection of *.contigs.fasta* files by the first letters of their respective filename, consisting of the sample's id number and abbreviated name, to make contigs identifiable in downstream analyses

Chapter: 2.5.

Input: a folder containing one or more *.contigs.fasta* files. Files will not be altered but a modified copy will be generated in a separate output folder

Execute: see script  
Dependencies: None.

12\_outerjoin.r

Purpose: merge two CSV files containing locus summary statistics of several samples into one, by performing an "outer join"

Chapter: 2.3.3. (not explicitly mentioned)

Input: two CSV files as exported from XLSX format, containing headers in their first line; the column to be used as basis for joining (i.e., the one with the locus names) must have the same header in both files (e.g., "locus"), while all other columns must have different headers in both files  
Caution: entries in the "locus" column must not contain individual sample IDs!

Execute: see script; only two files can be joined at a time, so the script has to be executed several times, using the result files as new inputs until all samples have been joined