File "4\_workflow\_commands.docx": BASH and AWK commands used in the study  
(and some command line parameters for tools used)

**General information:**

* **all commands are for use in a Linux command line; text given in square brackets [] is to be replaced with your own names, omitting the brackets. Similarly,** [path/to/folder\_containing\_xy] **is to be replaced by your own appropriate path to the respective file**
* **where no path but only a filename etc. is given in a command (like** ./1\_extract\_files.sh**), the command has to be executed in the folder containing the respective file (except in cases where e.g. a tool has been previously added to $PATH, see below). Also normally, when no explicit path is given, output from scripts or commands etc. (e.g.** mkdir -p prechoiceloci\_clusterconsensuses\_locuswise**) will be generated in the folder where the script is located / the command was executed**
* **It is strongly recommended to follow the naming schemes and conventions as provided here, as several commands and also scripts are tailored to them and errors might occur when using other names**
* **wherever scripts are involved, please find more detailed information in the help provided with the scripts, at Dryad and/or GitHub**
* **for each chapter in which BASH or other commands were used, the respective passages in the detailed methods document "1\_detailed\_methods.docx" (available at Dryad) are repeated in quotation marks, followed by commands and additional information**

2.3.1. Read processing and extraction of on-target reads

"Reads were demultiplexed using Guppy; from this point, analyses were performed separately for each individual. NanoFilt v.2.8.0 (De Coster et al., 2018) was used for length-filtering of reads shorter than 300 bp."

**Command for demultiplexing using Guppy:**

./guppy\_barcoder --input\_path /[path/to/folder]/pass --save\_path /[path/to/output\_folder] --config configuration.cfg -r --detect\_mid\_strand\_barcodes --min\_score\_mid\_barcodes 60.0 -t 3 --barcode\_kits EXP-PBC001 --trim\_barcodes --num\_extra\_bases\_trim 20

**Barcode sequences plus 20 additional basepairs are trimmed, and detection of mid-strand barcodes at a minimum score threshold of 60 is enabled.**

**Commands for cropping and length-filtering using NanoFilt:**

NanoFilt --headcrop 10 --tailcrop 10 --readtype 1D --logfile /[path/to]/logfile\_crop.txt < /[path/to/input].fastq > /[path/to/croppedoutput].fastq

NanoFilt -l 300 --readtype 1D --logfile /[path/to]/logfile\_300.txt < /[path/to/croppedoutput].fastq > /[path/to/filteredoutput].fastq

**NanoFilt is first used to clip 10 bp from the start and end of the reads to remove remnants of the adapter sequence detected with nanoQC, and then to filter reads with lengths shorter than 300 bp.**

"To determine on-target reads, the FASTA read files were BLASTed against the collection of source ESTs originally used to design the CompCOS capture probes (Mandel et al., 2014; hence referred to as "CompCOS ESTs"). The blastn command was executed using local BLAST+ v.2.12.0 (Altschul et al., 1990; Camacho et al., 2009) with parameters *-max\_target\_seqs* and *-max\_hsps* set to "1", resulting in a list which reports only the best hit of each query successfully BLASTed against the ESTs. No further filtering was applied. From the collection of all reads for each individual, those with a BLAST hit were then extracted with the faSomeRecords application (Kent utilities, download website see References)."

**Note that Step 2 has to be executed for each sample separately.  
Step 1: make BLAST database from probe source ESTs (it is assumed here that BLAST has been added to the $PATH of the system, if not, include the complete path file to makeblastdb and blastn):**

makeblastdb -in /[path/to]/COS\_sunf\_lett\_saff\_all.fasta -input\_type fasta -dbtype nucl -parse\_seqids -out CompCOS\_ESTs -title "CompCOS source ESTs"

**Step 2: BLAST the demultiplexed, length-filtered reads from NanoFilt against the CompCOS EST database (caution, blastn requires a FASTA file as input; convert the reads from FASTQ prior to executing this command):**

blastn -task megablast -query /[path/to]/BCXX\_minlen300.fasta -db /[path/to/BLAST/dbs]/CompCOS\_ESTs -out BCXX\_minlen300\_ontargetCompCOSESTs.txt -num\_threads 7 -max\_target\_seqs 1 -max\_hsps 1 -outfmt "6 qseqid qstart qend sseqid sstart send sstrand bitscore evalue pident length qcovhsp mismatch gapopen gaps"

**Note: "BCXX\_" in the input and output denotes the barcode number of the respective sample. The output is given in format "6" with several useful items of information included. Refer to local BLAST manual for further details on used parameters. The IDs from reads with a BLAST hit are then extracted from the result TXT file using awk:**

awk '{print $1}' BCXX\_minlen300\_ontargetCompCOSESTs.txt > BCXX\_minlen300\_ontargetCompCOSESTs\_ids.txt

**Based on this ID TXT file, the respective reads are then extracted from the original reads using the faSomeRecords application.**

2.3.2. Clustering and assignment of reads to their respective CompCOS locus

"During the clustering step, which was performed using VSEARCH v.2.15.0 (Rognes et al., 2016), appropriate adjustment of the clustering threshold parameter (CT) was of great importance."

**Command used for VSEARCH, to be executed separately for each sample sequenced:**

/[path/to]/vsearch-2.15.0-linux-x86\_64/bin/vsearch --cluster\_fast /[path/to]/ BCXX\_minlen300\_ontargetCompCOSESTs.fasta --centroids [name\_of\_centroids\_file].fasta --clusters vsearch\_088\_cluster --msaout [name\_of\_clusteralignments\_file].fasta --uc results\_uclust\_088.txt --blast6out results\_blast6\_088.txt --id [clustering\_threshold\_forexample\_0.88] --log [name\_of\_logfile].log --threads 8 --clusterout\_id --clusterout\_sort --consout [name\_of\_clusterconsensuses\_file].fasta --qmask none --strand both

**The command is executed for each desired CT separately (file naming should be changed accordingly). VSEARCH is started in *--cluster\_fast* mode, which ensures that the longest reads were preferably used as cluster centroids; this is important to avoid artificial splitting of otherwise uniform clusters. Each individual is assigned its own prefix for its clusters using the *--clusters* parameter (for example "vsearch\_088\_cluster"). No masking is performed (*--qmask none*). Setting *--strand* to *both* ensures that both strand directions are checked when comparing sequences with the cluster seed.**

"The partially overlapping raw reads within each cluster were then aligned using the local version of Lamassemble (Frith et al., 2021; see also Katoh et al., 2019). After alignment, all cluster files were renamed according to the best CompCOS BLAST hit of their centroids"

**The whole procedure given here, up to the renamed cluster files, is done separately for each sample sequenced. Step 1: prior to alignment, singleton clusters should be removed from the collection of clusters and moved to another folder. First, mark singleton clusters by adding an "\_s" to their filename:**

dir="[folder\_containing\_all\_clusters]";

files="$(ls $dir)";

for file in $files; do

count="$(grep -o '>' $dir/$file | wc -l)";

if [ "$count" == "1" ]; then

mv $dir/$file $dir/${file}\_s;

fi;

don**e;**

**Then, move them to another folder:**

ls /[path/to/folder\_containing\_all\_clusters]/\*\_s | xargs -n1 -I{} mv {} /[path/to/folder\_for\_singletons]/

**Step 2: align the reads within the remaining clusters using a BASH FOR loop and the lamassemble tool. Lamassemble is run with the alignment flag *-a* set and the *-m* and *-z* parameters according to the "accurate" preset in the online application at** [**https://mafft.cbrc.jp/alignment/server/index-rawreads.html**](https://mafft.cbrc.jp/alignment/server/index-rawreads.html)**, e.g., the *-m* parameter set to be two times the number of sequences contained in the largest cluster. The supplied LAST-generated *last-train* file "promethion.mat" is used:**

dir="[folder\_containing\_all\_clusters]";

files="$(ls $dir)";

for file in $files; do

lamassemble -a -g 50 -p 2e-3 -m [442] -z 1000 -P 8 /[path/to]/lamassemble/train/promethion.mat $dir/$file > /[path/to/folder\_containing\_aligned\_clusters]/$file;

done;

**Step 3: prior to renaming cluster files, the "\_s" suffix on singleton cluster filenames is removed again. Afterwards, singleton clusters are moved into the folder containing the aligned clusters (execute both commands within the "folder\_for\_singletons"):**

for i in \*\_s; do mv "$i" "${i/\_s/}"; done;

mv \* /[path/to/folder\_containing\_aligned\_clusters]/

**Step 4: for renaming loci, the results file from BLASTing the centroids (from VSEARCH) against the CompCOS ESTs, has to be manually modified in a text editor: the first semicolon of every line is replaced by a tab stop, to give each centroid ID its own column (column 1); also, the text "clusterid=" in each line is replaced by the prefix previously assigned to each cluster by VSEARCH with the *--clusters* parameter, so that column 2 now corresponds to the filenames of the clusters.**

**Step 5: the modified file is used as input for the following AWK command:**

cat [edited\_BLAST\_results\_file] | awk 'NF==16 {

a[$5]++

f\_source = "/[path/to/folder\_containing\_aligned\_clusters]/"$2

f\_dest = "/[path/to/folder\_containing\_aligned\_clusters]/BCXX\_"$5"\_"a[$5]"\_"$2

command = "mv "f\_source" "f\_dest

print command

system(command)

}'

**Caution: this will overwrite data, so make a backup of the aligned clusters folder first! Also, replace the number in "NF==16" in the code by the number of columns in your own file in case it is different! The "source" file (i.e., clusters file) for each renaming is taken from column 2 (= filename) of the BLAST results file, and the "destination" filename is composed of a prefix = sample ID (must be in the format "BCXX", replace XX with your respective barcode number!), and the content of column 5 in the BLAST results file (best CompCOS EST hit found for the read in the BLAST search), plus the original cluster filename at the end ("$2"). As most ESTs received more than one cluster, an additional array is incorporated, adding a sequential number after each EST to the filename. The procedure results in cluster filenames in the format "sample\_id - EST\_name - 1 - previous\_cluster\_filename". Be sure to change the prefix manually in the above command for each sample processed. An analogous renaming procedure can be performed using the BLAST results file for the VSEARCH cluster consensuses instead of the centroids.**

2.3.3. Loci summary statistics and pre-choice loci

"In order to enable a thorough examination of the loci, and also for calculation of the first of the four indices (see below), loci summary statistics on the clustering of each sample and across all individuals were calculated first"

**Calculation of total number of reads per enriched locus, to be done separately for each sample:**

for prefix in $(ls /[path/to/folder\_containing\_aligned\_clusters]/ | cut -c1-14 | uniq); do

x=$(ls /[path/to/folder\_containing\_aligned\_clusters]/${prefix}\* | xargs -n1 grep -o ">" | wc -l)

echo "${prefix} ${x}"

done > readsperlocus.txt

**Note that "cut -c1-14" extracts the first 14 characters from the filenames, which here correspond to the sample ID plus the locus name. This number might have to be adjusted for other capture probe sets. The same code can be applied to a folder containing only singleton clusters (see above) to calculate the amount of singleton clusters per locus (as each singleton cluster comprises one single read only). Optionally: two statistics TXT files created in this way can be joined using the "join" command:**

join -a1 <(sort -k1 readsperlocus.txt) <(sort -k1 singletonsperlocus.txt) > combined\_statistics.txt

**Calculation of total number of clusters per locus, to be done separately for each sample:**

for prefix in $(ls /[path/to/folder\_containing\_aligned\_clusters]/ | cut -c1-14 | sort | uniq); do

x=$(ls /[path/to/folder\_containing\_aligned\_clusters]/${prefix}\* | wc -l)

echo "${prefix} ${x}"

done > clusterperlocus.txt

**Note that "cut -c1-14" may require changing, see above. For the CompCOS loci probe set, "cut -c1-18" can also be applied to obtain the number of clusters per EST, if desired. Values within all statistics TXT files can be transferred into an Excel sheet to infer further statistical values, here named the sample-wise loci summary statistics file (see file "6\_full\_locus\_statistics.xlsx" at Dryad). For example, by subtracting the amount of singleton clusters (as obtained above) from the total number of clusters, the number of non-singleton clusters per locus can be deduced.**

**Calculation of number of clusters with at least five (or ten) reads, to be done per locus and separately for each sample: First, calculate number of reads in each cluster:**

dir="[path/to/folder\_containing\_aligned\_clusters]";

files="$(ls $dir)";

for file in $files; do

count="$(grep -o '>' $dir/$file | wc -l)";

echo "${file} ${count}"

done > readspercluster.txt

**Then, find in the resulting TXT file the clusters with at least five (or ten) reads (replace "$2 >4" by "$2 >9" for clusters with 10+ reads) and sum them up for each locus; execute in the folder that contains "readspercluster.txt":**

awk '{

if ($2 >4)

print substr($1, 1, 14);

}' readspercluster.txt | sort | uniq -c > clusterwithmin5reads\_perlocus.txt

**Note that "substr($1, 1, 14)" again corresponds to sample ID + locus in the original filenames, analogous to "cut -c1-14" (see above). Results from these TXT files are added to the sample-wise loci summary statistics XLSX file as well (file "6\_full\_locus\_statistics.xlsx"); as "clusterwithmin5reads\_perlocus.txt" will not contain data for each locus, an easy way to make both files "compatible" would be to export from the XLSX file the columns containing locus names and non-singleton clusters as a TXT file (with spaces set as separator symbol) and to use that as template to join with "clusterwithmin5reads\_perlocus.txt" using the "join" command as described above. Note that the locus names have to be identical in both files for "join" to work properly:**

join -1 1 -2 2 -a1 <(sort -k1 [txtexportedfromxlsx.txt]) <(sort -k2 clusterwithmin5reads\_perlocus.txt) > clusterwithmin5reads\_perlocus\_formatted.txt

**The sample-wise loci summary statistics XLSX files are then successively joined into one single loci summary statistics file using the helper script 12\_outerjoin.r. The information contained in the final XLSX file can be used to calculate further relevant statistics, like sums of clusters and reads per locus across all samples, and also percentages of non-singleton, low-coverage and lower-coverage clusters.**

"Consequently, as a first approximation, loci with fewer than 100 and more than 1,000 sequenced reads were excluded. Furthermore, only loci with reads present in three or more individuals and with at least one cluster (per individual) containing five or more reads were kept. Clusters from these "pre-choice loci" were extracted for each individual (script 1\_extract\_files.sh) from its respective folder containing all aligned, renamed cluster files (result of 2.3.2.) after singleton clusters (i.e., clusters with only one read) had been removed from the folders."

**Step 1: for each sample/individual ("indX", change to your own sample number), in the folder containing all aligned, renamed cluster files, mark singleton clusters by adding an "\_s" to their filename (see also above):**

dir="[indX\_input\_folder]";

files="$(ls $dir)";

for file in $files; do

count="$(grep -o '>' $dir/$file | wc -l)";

if [ "$count" == "1" ]; then

mv $dir/$file $dir/${file}\_s;

fi;

done;

**Remove singleton clusters to another folder (create this folder first!):**

ls /[indX\_input\_folder]/\*\_s | xargs -n1 -I{} mv {} /[indX\_singleton\_cluster\_folder]/

**Step 2: in the loci summary statistics file, create a new sheet only containing entries for pre-choice loci (see file "6\_full\_locus\_statistics.xlsx", sheet "pre\_choice\_loci"); from this sheet, copy the first column (locus names, here in format as, e.g., "At1g01050") into a new CSV document (make sure to include UNIX line breaks!)**

**Step 3: use script 1\_extract\_files.sh to extract the clusters from the loci included in the list to another folder (must be present already!) using this command, repeat for each sample; result is one folder with pre-choice loci per sample:**

./1\_extract\_files.sh [name\_of\_csv\_file].csv [indX\_input\_folder] [indX\_prechoice\_loci\_folder]

**Note: input and output folder as well as the CSV must be in the same folder! Otherwise provide full paths to each location.**

"Low-coverage clusters then also were removed from the pre-choice collections (see the Discussion of the paper). For the remaining clusters, their consensus sequences were calculated from the alignments via lamassemble using the consensus flag *-c* and setting *-g* to 50."

**Step 1: add to the filename of each cluster the number of reads it contains, do for each sample**

dir="[indX\_prechoice\_loci\_folder]";

files="$(ls $dir)";

for file in $files; do

count="$(grep -o '>' $dir/$file | wc -l)";

mv $dir/$file $dir/${file}\_${count}reads;

done;

**Step 2: remove low-coverage clusters < 5 reads from the collections to another folder (repeat the command below for "\_2reads", "\_3reads" and "\_4reads"! singleton clusters have been removed previously)**

ls /[indX\_prechoice\_loci\_folder]/\*\_2reads | xargs -n1 -I{} mv {} /[indX\_lowcov\_cluster\_folder]/

**Step 3: for all remaining aligned cluster files within "indX\_prechoice\_loci\_folder", create the consensus sequence of each cluster using lamassemble and a FOR loop (repeat for each sample):**

dir="[indX\_prechoice\_loci\_folder]";

files="$(ls $dir)";

for file in $files; do

lamassemble -c -g 50 -p 2e-3 -n $file -m 720 -z 1000 -P 5 /[path/to]/lamassemble/train/promethion.mat $dir/$file > /[path/to]/prechoiceloci\_clusterconsensuses/${file}\_cons;

done;

**The command unites all generated cluster consensus files into one single folder "prechoiceloci\_clusterconsensuses". Also, each consensus sequence is given the name of its respective input file ("-n $file"), which would thus have the format "sample\_id - EST\_name - 1 - previous\_cluster\_filename - number of contained reads". As alignment of reads has been done previously and only the consensus sequences are generated here (*-c* flag), the parameters *-P*, *-m* and *-z* should have no function and could be omitted.**

"Consensus sequences of all individuals were combined locus-wise into common FASTA files. These FASTA files served as input for the scripts calculating indices 2-4 (see below).**"**

**The FOR loop is executed in folder "prechoiceloci\_clusterconsensuses", where the consensus sequence files are located. It generates a new folder "prechoiceloci\_clusterconsensuses\_locuswise" with one file per locus, containing locus-wise concatenated consensus sequences from all samples:**

mkdir -p prechoiceloci\_clusterconsensuses\_locuswise

rm -rf prechoiceloci\_clusterconsensuses\_locuswise/\*

for infile in BC\*; do

outfile=$(echo $infile | cut -c 6-14)

cat $infile >> prechoiceloci\_clusterconsensuses\_locuswise/$outfile

done

**All files with filenames starting with "BC" are processed. Note again that "cut -c 6-14" extracts the locus name from the filename of the consensus files and might need to be changed for other naming schemes or probe sets.**

2.3.4. Choice of putatively non-paralogous loci, based on four specialized indices

"The script 4\_index3entropy\_index4silhouette.r uses the same input as the script for obtaining the KBS index, and was executed in a loop over all FASTA files of the respective folder. As output, it produces a CSV file with the coefficients alongside a graphical representation in PDF format for each locus. Apart from that, it also provides a summary of all information for the respective input locus (silhouette coefficients, best value of *K*, cluster membership frequencies of each taxon for the best value of *K*, and mean entropy, see above). For convenience, the best *K* and entropy values of all loci were finally united in a single CSV file."

**Step 1: as script 4\_index3entropy\_index4silhouette.r is intended for one single FASTA file, it is here executed with all locus-wise concatenated consensus sequences as input in a FOR loop. Before execution, copy the script into the input folder "prechoiceloci\_clusterconsensuses\_locuswise" and execute the command from within the folder  
(all files here are named by their respective locus hence are supposed to begin with "At");**

for f in At\*; do

/[path/to]/prechoiceloci\_clusterconsensuses\_locuswise/4\_index3entropy\_index4silhouette.r -k 8 -l single -o ${f/\_\*/} $f > ${f/\_\*/}.txt

done

**this will save the screen output for each locus containing the summary to a TXT file; all output will be saved in the folder where the script is located. For more information on parameters, refer to the script!**

**Step 2: summarize best *K* and entropy information for all loci from TXT files to one CSV file (execute in same folder!):**

for f in \*.txt; do

loc=${f/.txt/}

k=$(awk '/^best/ {print $5}' $f)

H=$(awk '/^entropy/ {print $2}' $f)

echo -e "$loc\t$k\t$H"

done > summary\_bestK\_entropy\_prechoice\_loci.csv

**The results can then be added to the loci summary statistics file.**

"The selected candidate loci were then extracted from the collection of pre-choice loci FASTA files (see 2.3.3.) with script 1\_extract\_files.sh."

**in the loci summary statistics file, create a new sheet only containing entries for the proposed candidate loci (see file "6\_full\_locus\_statistics.xlsx" at Dryad, sheet "candidate\_loci"); from this sheet, copy the first column (locus names, here in format e.g. "At1g01050") into a new CSV document (make sure to include UNIX line breaks!)**

**use script 1\_extract\_files.sh to extract the FASTA files from the loci included in the list to another folder (must be present already!) using this command**

./1\_extract\_files.sh /[path/to]/[csv\_file\_with\_candidate\_loci\_names].csv /[path/to]/prechoiceloci\_clusterconsensuses\_locuswise /[output/path/to]/candidate\_loci

2.3.5. Assessment of candidate loci

"Furthermore, in each locus's FASTA file (with cluster consensus sequences of all individuals), the sequences were aligned alongside the respective source ESTs using again lamassemble with settings for alignment as mentioned above, plus the *--all* flag set."

**Create a new folder named "candidate\_loci\_aligned" first. Note that source EST FASTA files should be merged with their respective locus FASTA files prior to executing the FOR loop. Alignment can certainly also be done without added ESTs.**

dir="/[path/to/]/candidate\_loci";

files="$(ls $dir)";

for file in $files; do

lamassemble -a --all -g 50 -p 2e-3 -m 56 -z 1000 -P 5 /[path/to]/lamassemble/train/promethion.mat $dir/$file > /[path/to]/candidate\_loci\_aligned/ ${file}\_ESTs;

done;

**The *-m* parameter is again set to be twice the largest number of sequences contained in any of the locus files; see above and modify as needed. The *-all* parameter is set here to avoid truncation of regions with only one sequence at the beginning or end of alignments.**

2.4.3. Data pre-processing

"… adequately dealing with possible chimeric sequences was of particular importance and was incorporated into the workflow at several steps (Dryad file "3\_detailed\_workflow2\_ampliconSeq.pdf", "*Filter Chimeras! #1-4*"). Prior to demultiplexing, all reads were mapped to reference sequences of all loci (preparation of references see below), and whenever a read mapped to two loci, it was dismissed. Reads producing supplementary mappings were also excluded in this step. Mapping was executed once for each library, using Minimap2 v.2.21 (Li, 2018; Li, 2021); filtering was done using BASH commands and filterbyname.sh from BBTools."

**Step 1: for this FOR loop, the input folder is the one containing the mapping references of all amplified loci (caution: only one single reference FASTA file per locus!). The input file is the FASTQ file containing all reads of a library after length filtering by NanoFilt (here called "reads\_min500max3800.fastq"). The output folder must be created before (here called "chimera\_removal\_2"). Appropriate paths to Minimap and Samtools binaries must be given, according to the installed program version. Output from mapping is piped to Samtools and output as sorted BAM file; for this, a folder for intermediate files must be specified (here called "mytemp"). For further parameters refer to the help of the respective tools.**

dir="/[path/to]/mappingreferences";

files="$(ls $dir)";

for file in $files; do

/[path/to]/minimap2-2.21\_x64-linux/minimap2 -ax map-ont -t 8 $dir/$file /[path/to]/reads\_min500max3800.fastq | /[path/to]/samtools-1.13/samtools sort -o /[path/to]/chimera\_removal\_2/${file}\_minimap.sorted.bam -T /[path/to]/mytemp

done;

**Step 2: the output files also contain lines referring to unmapped reads or secondary mappings; these have to be removed before mapping files can be compared across loci (caution: this loop as well as some of the commands below assume that Samtools has been added to the $PATH of the system. Add its full path if necessary!):**

dir="/[path/to]/chimera\_removal\_2";

files="$(ls $dir)";

for file in $files; do

samtools view -h -F 4 -F 256 -o /[path/to]/chimera\_removal\_2/`echo $file | cut -c-12`\_primarysupplementarymapped.sam $dir/$file

done;

**This filtering step also excludes reverse complemented secondary mappings ("-F 272"). Output filenames are shortened to their locus name plus a suffix for sake of readability (again using "cut", see above); adjust number of extracted letters if necessary. In this step, the BAM files are also converted to SAM to make them human-readable.**

**Step 3: the first column of each SAM file (containing the read IDs) is extracted into a TXT file list (please move all BAM files from folder "chimera\_removal\_2" to another location before executing this step):**

dir="/[path/to]/chimera\_removal\_2";

files="$(ls $dir)";

for file in $files; do

awk '{print $1}' $dir/$file > ${file}\_ids.txt

done;

**The output is generated in the folder the loop was executed in.**

**Step 4: in all output files, extracted lines referring to the header of the original SAM files (something like "@HD", "@SQ" or "@PG") must be manually deleted before proceeding.**

**Step 5: all "\_ids.txt" files are then concatenated (execute in the folder containing the "\_ids.txt" files), and the resulting file searched for duplicate read IDs:**

cat \*\_ids.txt > allids.txt

cat allids.txt | sort | uniq -d > duplicate\_readids\_mappingchimdetect.txt

**Note that the resulting list will also include reads with supplementary mappings, as duplicates among two or more loci, but also within a single locus will be identified.**

**Step 6: the reads with the IDs NOT contained in the result file can then be extracted from the original FASTQ file containing all reads, using filterbyname.sh from BBTools:**

/[path/to]/bbmap/filterbyname.sh in=/[path/to]/reads\_min500max3800.fastq out=/[path/to]/reads\_min500max3800\_filtered.fastq substring=name ow=f names=/[path/to]/duplicate\_readids\_mappingchimdetect.txt ignorebadquality=t qin=33 qout=33 -da

**Be sure to include all parameters, as otherwise quality values of the reads might not be preserved correctly. The output file "reads\_min500max3800\_filtered.fastq" is then demultiplexed using the Guppy barcoder.**

**Optionally: if statistics on total mapped reads per locus for a library are desired, the mapping files containing primary and supplementary mapped reads ("\_primarysupplementarymapped.sam") can be analyzed using Samtools flagstat. Therefore, all SAM files have to be converted to BAM format first:**

dir="/[path/to]/chimera\_removal\_2";

files="$(ls $dir)";

for file in $files; do

samtools view -b -h -o ${file}.bam $dir/$file

done;

**Statistics via Samtools flagstat (execute for each BAM file separately, in their parent folder; or include in a FOR loop as above):**

/[path/to]/samtools-1.13/samtools flagstat [locusname]\_primarysupplementarymapped.sam.bam

**Statistics can also be extracted from the "\_minimap.sorted.bam" files containing unmapped reads and secondary mappings.**

2.4.4. Mapping and filtering

"Demultiplexed reads from each individual were then mapped to all loci using Minimap2."

**Step 1: prior to mapping, FASTQ files from Guppy demultiplexing have to be copied from their barcode folders to one common folder named "reads" and be renamed according to the respective sample (here e.g. "16\_208\_01\_Lhall.fastq", = running number "16", sample ID "208-01", and sample name "Lhall", *Leucanthemum halleri*). Note: ensure that all names are chosen following the same structure with identical name lengths (e.g., sample name always shortened to five letters) to enable error-free downstream processing. In this study, one of the libraries was run on two flongle flowcells, which resulted in two FASTQ files per barcode; these were concatenated.**

**Step 2: the reference FASTAs for mapping (one file per locus, containing one or more sequences, prepared as described in chapter 2.4.4.) have to be placed into one folder "mappingreferences". Their filenames consist of a running number plus the locus name from the source ESTs plus a suffix "\_Cons" added after consensus generation, e.g. "44\_At1g68570\_Cons.fasta".**

**Step 3: a folder "results" has to be generated manually prior to executing a double FOR loop to do the actual mapping with Minimap2:**

dir="/[path/to]/mapping/reads";

dir2="/[path/to]/mapping/mappingreferences";

reads="$(ls $dir)";

refs="$(ls $dir2)";

for indiv in $reads; do

for ref in $refs; do

/[path/to]/minimap2-2.21\_x64-linux/minimap2 -ax map-ont -t 8 $dir2/$ref $dir/$indiv > /[path/to]/mapping/results/`echo ${indiv} | cut -c 1-15`\_locus`echo ${ref} | cut -c 1-2`\_minimap.sam

done

done;

**This will consecutively map reads from each individual sample to all references provided, with the result files saved into folder "results". Results filenames consist of the first 15 letters of the filenames in the "reads" folder (basically, the name without the FASTQ suffix; "`echo ${indiv} | cut -c 1-15`"), and a prefix "\_locus" followed by the first two letters of the filenames in the "mappingreferences" folder (the locus' running number; "`echo ${ref} | cut -c 1-2`") plus a suffix "\_minimap.sam", summing up to e.g. "16\_208\_01\_Lhall\_locus44\_minimap.sam". Change to own preferences if needed, however a consistent naming scheme should be applied. Note: it is advisable to add MD tags to the SAM files by adding the *--MD* parameter to the Minimap2 command! However, this has not been done in the present study (see below).**

"An additional step was incorporated to assess the quality of the mapping and the used references, by extracting for each individual the reads that remained unmapped over all loci (script 8\_filter\_entirelyunmappedreads.sh), and locally BLASTing those reads against the mapping references of the loci, with settings as applied in 2.3.1."

**The script 8\_filter\_entirelyunmappedreads.sh produces, apart from TXT files with the respective read IDs, one FASTQ file per individual, containing the reads that did not map to any of the references. These are converted to FASTA format and then BLASTed against all reference sequences; for use of makeblastdb and blastn see above. Step 1: generate BLAST database from mapping reference FASTAs (concatenate all of them into one file first!):**

makeblastdb -in /[path/to/all\_reference\_FASTAS\_concatenated].fasta -input\_type fasta -dbtype nucl -parse\_seqids -out allmappingreferences -title "CompCOS xy loci references"

**Step 2: locally BLAST entirely unmapped reads from all individuals (files in folder "reads\_unmapped" according to script 8\_filter\_entirelyunmappedreads.sh, named according to the sample filename scheme "running number - sample ID - sample name" plus a suffix "\_unmapped", e.g. "16\_208\_01\_Lhall\_unmapped.fasta") against all mapping references:**

dir="/[path/to]/reads\_unmapped";

files="$(ls $dir)";

for file in $files; do

blastn -task megablast -query $dir/$file -db /[path/to/BLAST/dbs]/allmappingreferences -out /[path/to]/BLASTresults/`echo ${file} | cut -c 1-15`\_BLASTtoreferences.txt -num\_threads 8 -max\_target\_seqs 1 -max\_hsps 1 -outfmt "6 qseqid qstart qend sseqid sstart send sstrand bitscore evalue pident length qcovhsp mismatch gapopen gaps"

done;

**The loop produces a collection of TXT files with BLAST hits, separately for each sample, in the folder "BLASTresults". Output filenames carry the 15 first letters of the input FASTAs (i.e., complete except the "\_unmapped" suffix) plus the defined new suffix "\_BLASTtoreferences.txt". Refer to local BLAST manual for further details on used parameters.**

**Step 3: analysis of the results was done manually: the number of lines in the TXT file of unmapped read IDs (output from script 8\_filter\_entirelyunmappedreads.sh) equals the amount of entirely unmapped reads in a given sample, so that percentages of unmapped reads can be inferred. The number of hits in the BLAST result files was used to calculate the percentage of hits relative to the number of entirely unmapped reads in each sample. Based on this information, samples were determined for which preparation of a specific mapping reference seemed appropriate.**

"In a next step, the mappings were evaluated regarding numbers of primary, secondary and supplementary alignments. … Evaluation of mappings was done to identify possible problems during mapping, using the Integrative Genomics Viewer (IGV) v.2.4.15 (Robinson et al., 2011), Samtools flagstat and the script 9\_extract\_mapping\_statistics.sh on the original data and after each monitoring / filtering step. First, mappings with <100 reads were visually inspected to identify possible reasons for the poor mapping performance; where reasonable, additional mapping references were built using Canu and the respective mapping repeated. Mappings with zero to six reads were then removed from the collection. The remainder was subjected to a filtering process where first, unmapped, secondary and supplementary mapping entries were removed from the SAM files manually or using Samtools view. Then, another chimera filtering step followed (#4): … mappings were first filtered for diagnostic purposes only, using a low threshold for soft-clipped bases (*clipfilter=50*) in the reformat.sh script (BBTools); no reads were excluded at this step. Mappings losing >50% of their reads during this tentative filtering were again examined manually, and suitable new mapping references generated where necessary via Canu. In a second step, reads in mappings were filtered using a much more conservative threshold (*clipfilter=300*)."

**Step 1: Create mapping statistics  
execute Samtools flagstat on all SAM files resulting from the mapping procedure, via a FOR loop, which will save the resulting statistics files, named according to the respective input file plus a "\_flstat" suffix, into a folder "results\_stats" (must be manually created first!)**

dir="/[path/to]/mapping/results";

files="$(ls $dir)";

for mapping in $files; do

samtools flagstat $dir/$mapping > /[path/to]/results\_stats/`echo ${mapping} | cut -c 1-25`\_flstat

done;

**note: Only the first 25 characters of the input file name will appear in the output file name ("`echo ${mapping} | cut -c 1-25`")**

**then, use script 9\_extract\_mapping\_statistics.sh to summarize the information from the many "\_flstat" files into a single CSV; information on primary mapped, secondary and supplementary mapped reads will be extracted.**

**This combination of loop plus shell script was executed on the original mappings, mappings after removal of unmapped, secondary and supplementary mapping entries (for checking purposes), mappings after filtering softclipped reads (both settings), and whenever mappings had to be repeated due to production of a new, custom mapping reference (see underlined text in the paragraph from file "1\_detailed\_methods.docx" cited above). The results in all "mapping\_statistics.csv" files generated by script 9 were combined into what is available as file "8\_mapping\_and\_canu\_stats.xlsx" at Dryad.**

**Step 2: Filtering process 1  
Remove entries regarding unmapped reads, secondary and supplementary mappings from the SAM files. The loop again assumes that Samtools has been added to the $PATH of the system.**

dir="/[path/to]/mapping/results";

files="$(ls $dir)";

for mapping in $files; do

samtools view -h -F UNMAP,SECONDARY -o /[path/to]/mapping/results\_postfilter1/${mapping} $dir/$mapping

done;

**For the resulting SAM files, new mapping statistics can be created (see above); these results were added to file "8\_mapping\_and\_canu\_stats.xlsx", column "post-filter 1". Supplementary reads contribute two lines to the mapping files, one with SAM flag *2048* (or *2064* when reverse-complemented) and another with SAM flag *0* (or *16*). As such reads were to be excluded completely, all respective lines were removed manually in the SAM files.**

**Step 3: Filtering process 2 - 50 bp**

**Using BBTools reformat.sh and a FOR loop, reads with >50 bp softclipped bases are trimmed from the mappings and the results evaluated. Note: the original mappings remain unchanged at this step and the resulting new mapping files are not used further! The output folder "results\_postfilter2\_50bp" has to be manually created first.**

dir="/[path/to]/mapping/results\_postfilter1";

files="$(ls $dir/\*locusXX\*)";

for mapping in $files; do

name=$(basename $mapping)

/[path/to]/bbmap/reformat.sh in=$mapping out=/[path/to]/mapping/results\_postfilter2\_50bp/${name} ref=/[path/to]/mapping/mappingreferences/[name\_of\_mapping\_reference\_of\_locusXX].fasta clipfilter=50 mincalledquality=0 maxcalledquality=99

done;

**Unfortunately, in the study, original mapping of amplicon reads to references (see above, 2.4.4.) was done without adding MD tags to the resulting SAM files. This means that the reference sequence could not be deduced from the SAM files by reformat.sh, which is why it had to be given in the command ("ref="). In consequence, the loop had to be executed separately for each locus; SAM files for each locus were extracted from folder "results\_postfilter1" by the respective locus' running number plus the prefix "locus" (filename format in folder "mappingreferences", see 2.4.4.); "\*locusXX\*" acts as placeholder for these files, where XX corresponds to the locus' running number. The corresponding mapping reference file is then added via the *ref* parameter, and the loop repeated for each locus. With MD tags added to SAM files during the original mapping procedure, the *ref* parameter could possibly be dismissed and all input files processed within the same loop ("files="$(ls $dir)";"). Mapping statistics for this step were added to file "8\_mapping\_and\_canu\_stats.xlsx", column "post-filter 2 (filter reads with softclipped bases; 50 bp evaluation)".**

**Step 4: Filtering process 2 - 300 bp  
The mappings resulting from step 2 are filtered again using the loop from step 3, but setting *clipfilter=300* and directing the output to another folder "results\_postfilter2\_300bp". These SAM files are further processed; their mapping statistics were added to file "8\_mapping\_and\_canu\_stats.xlsx", column "post-filter 2 (filter reads with softclipped bases; 300 bp)".**

2.4.5. Read extraction and *de novo* assembly

"From the filtered mappings, the reads were extracted with Samtools fastq and reads were assembled *de novo* for each individual and locus using Canu"

**After removal of unmapped reads, secondary and supplementary mappings as well as putative chimeric reads, the remaining reads could be extracted from the SAM files for subsequent *de novo* assembly:**

dir="/[path/to]/results\_postfilter2\_300bp";

files="$(ls $dir)";

for mapping in $files; do

samtools fastq $dir/$mapping > /[path/to]/extractedreads/`echo ${mapping} | cut -c 1-25`.fastq

done;

**Result filenames were once more reduced, to represent the running number, ID and name of the sample, as well as the ID of the locus the reads belonged to (e.g. "16\_208\_01\_Lhall\_locus44.fastq"; realized via "`echo ${mapping} | cut -c 1-25`.fastq").**

**Assembly is done in a FOR loop processing all FASTQ files belonging to the same locus (extracted again via providing the respective locus' running number in "ls $dir/\*locusXX\*"). The loop has to be repeated for each locus manually (caution: replace "XX" appropriately also throughout the command!). A folder "Canu" contains all results and has to be manually generated beforehand, as well as a subfolder named according to the running number ("XX") of the locus currently processed.**

dir="/[path/to]/extractedreads";

files="$(ls $dir/\*locusXX\*)";

for reads in $files; do

name=$(basename $reads | cut -c 4-23)

mkdir /[path/to]/Canu/XX/$name

/[path/to]/canu-2.1/bin/canu -p $name -d /[path/to]/Canu/XX/$name/ -nanopore $reads genomeSize=[] minReadLength=[] minOverlapLength=[] readSamplingBias=0.0 readSamplingCoverage=400 correctedErrorRate=0.134 corOutCoverage=9000 corMhapSensitivity=low obtMhapSensitivity=low utgMhapSensitivity=low stopOnLowCoverage=50 contigFilter="1 0 1.0 0.5 0"

done;

**The command generates a new subfolder in folder "XX" for each file (= each sample) processed ("mkdir /[path/to]/Canu/XX/$name"). "$name" is set to be the filename of the respective read FASTQ file, shortened to only the sample ID, sample name and locus running number (e.g. "208\_01\_Lhall\_locus44"), via "name=$(basename $reads | cut -c 4-23". "$name" is also used as the prefix for the respective Canu output files. Make sure to follow these naming schemes to ensure smooth downstream processing.**

**Standard settings for Canu were as follows (these were modified where necessary, see file "1\_detailed\_methods.docx" at Dryad as well as the Results section of the paper): automatic downsampling of reads before loading them into the sequence store was set to an amount equalling 400x read coverage of the marker (*readSamplingCoverage=400*) in contrast to recommendations by the software authors, in order to keep sufficient amounts of reads available for assembly of all potential paralogs. No positive read sampling bias was applied (*readSamplingBias=0*) in order not to privilege left-over long chimeric reads. Consequently, the shortest reads, which were not supposed to enter assembly, had to be filtered using the *minReadLength* parameter. The latter as well as *minOverlapLength* was set locus-wise, to 2/3 of the length of the shortest mapping reference used; the genome size was set to the length of the longest reference. For correction and assembly of all reads present in *seqStore*, *corOutCoverage* was set to a very high value (*9000*). The settings for the *correctedErrorRate* and the three *MhapSensitivity* parameters were chosen as recommended for high-coverage datasets by the online Canu parameter reference. Canu was also set to abort whenever read coverage dropped below 50x at any time prior to assembly. Only very limited contig filtering was applied (*contigFilter="1 0 1.0 0.5 0"*). These settings were basically also used in cases where specific references had to be created to improve the original mapping procedure (see file "1\_detailed\_methods.docx", chapter 2.4.4.).**

2.5. Marker alignments and inference of phylogenetic trees

"After assembly, the resulting contig FASTA files were extracted from their results folders and the sequence names of the contigs within (default "tig00000001", "tig00000002") changed to include the respective sample's ID number and abbreviated name, alongside a sequential number where necessary. This was done by the script 11\_sequencename\_from\_filename.sh."

**Step 1: from all Canu results folders, all *.contigs.fasta* files are copied into a new directory called "contigs", using a FOR loop. The folder structure is expected to be as was set up in the previous chapter: one folder ("Canu"), containing a collection of subfolders (one for each amplified locus), each with again subfolders for each sample, generated during Canu assemblies (see above):**

for contig in \*/\*/\*contigs.fasta; do

cp -a $contig /[path/to]/contigs/

done;

**Step 2: then, using the script 11\_sequencename\_from\_filename.sh, replace all SEQUENCE names of contigs within the *.contigs.fasta* files by the first 12 characters of their respective filename. Filenames here are in the format "sample ID - abbreviated sample name - locus running number", e.g. "208\_01\_Lhall\_locus44.contigs.fasta" (see above), so extraction of the first 12 characters results in sequence names of the structure "208\_01\_Lhall\_1"; the script will also add a sequential number (e.g. "\_1") to each sequence name, to take into account cases with more than one contig per file.**

"FASTA files were then concatenated locus-wise using *cat*, and aligned with MAFFT v.7.490."

**The *--threadit* parameter is set to *0* in order to obtain reproducible results; automatic method selection is enabled (*--auto*) as well as adjustment of sequence input direction (*--adjustdirection*); this is important as contigs might be assembled reverse-complemented.**

mafft --anysymbol --adjustdirection --inputorder --auto --thread 4 --threadtb 4 --threadit 0 [input\_file].fasta > [output\_file].fasta

"Gene trees were calculated for all markers using IQ-TREE v.1.6.12 (Nguyen et al., 2015; Kalyaanamoorthy et al., 2017; Hoang et al., 2018)."

**Parameter settings for IQ-TREE include repeated automatic model selection for every run (*-mredo*), more thorough NNI search turned on (*-allnni*), 1,000 ultrafast bootstrap replicates (*-bb 1000*) and 3,000 iterations for individual bootstrap trees done (*-nm 3000*), and near-zero-length branches collapsed in the final tree (*-czb*).**

/[path/to]/iqtree-1.6.12-Linux/bin/iqtree -s [input\_alignment].fasta -st DNA -o [outgroup\_sequence\_name] -pre [results\_prefix] -redo -nt AUTO -v -allnni -bb 1000 -wbtl -nm 3000 -m MFP -mredo -minsup 0.5 -bi 100 -con -czb