HybSeq course — from raw data to species trees

Practical processing of HybSeq target enrichment sequencing data on computing grids like MetaCentrum — pre-processing, HybPiper, alignments, gene trees, species trees

Tomáš Fér, Roswitha Elisabeth Schmickl and **Vojtěch Zeisek**

Department of Botany, Faculty of Science, Charles University, Prague
Institute of Botany, Czech Academy of Sciences, Průhonice
https://trapa.cz/, zeisek@natur.cuni.cz

June 10–18, 2024

Outline I

Target enrichment for plant/animal systematics — methodological workshop (MB120C117)

Outline II

Target enrichment for plant/animal systematics — methodological workshop (MB120C117)

Outline III

Target enrichment for plant/animal systematics — methodological workshop (MB120C117)

Resources before we start

- Course Git (slides with all links) and information in SIS (česky)
- Scripts `https://github.com/V-Z/hybseq-scripts` **shown in the course** — clone the Git repository and adapt to your needs
- Download presentation from `https://trapa.cz/en/hybseq-course-2024`
- Most of the work is done in Linux/UNIX (macOS, ...) command line, so that good knowledge of work in command line is essential, good starting point can be my Linux and MetaCentrum course `https://soubory.trapa.cz/linuxcourse/`
- Many tasks are done in R, so that at least basic knowledge of R is needed, good starting point can be my R course `https://soubory.trapa.cz/rcourse/`
- Processing HybSeq data is computationally demanding (it requires plenty of resources), during the course we use MetaCentrum, Czech National Grid Infrastructure (česky) (slide 14), but any computing cluster or powerful desktop (for patient users;-) can be used

HybSeq and its data

- HybSeq combines target enrichment and genome skimming (see lesson by RS) and especially in lager plant genomes it allows to efficiently select only ∼1000 single/low copy genes
- It requires sequencing probes, general or group specific (can be design using pipelines like Sondovač)
- From sequencing laboratory we get demultiplexed raw FASTQ files
- Steps leading to lists of gene trees require plenty of computing resources and disk space
  - Even simple operation can take significant time — think twice before every step
  - User can select how much resources provide for each step — depends on data size and available resources (more resources like CPU and memory will speed up processing)

Zingiberaceae test data set

- Family Zingiberaceae has altogether ca. 1600 species throughout tropical Africa, Asia, and the Americas
- We selected 35 members from TF's dataset as test data
- The probes used for sequencing were introduced in Fér et Schmickl 2018 (reduced to 250 probes)
- For data structure see slide 21
- If you did not yet do so, download data (slide 24) — it can take some time...

Steps from sequencing files to species trees I

1. Trimming of raw sequencing FASTQ files (removal of adaptors, ...) e.g. by Trimmomatic
2. Deduplication of FASTQ reads e.g. by BBMap
   - Not strictly required, duplicates mainly provide wrong insight into real coverages of loci
3. Checking of FASTQ files in FastQC or similar tool and removal of low-quality files
4. Preparing probe reference FASTA file and list of samples for processing by HybPiper
5. Processing every sample with with HybPiper (or alternatively HybPhyloMaker — see lessons by TF — or similar tool)
   1. Mapping of FASTQ reads with BWA to FASTA reference
   2. Distributing (sorting) of reads according to successful hits (using Samtools) into FASTA files for assembly
   3. Assembly of sorted reads with SPAdes
   4. Alignment of SPAdes contigs against the target sequence
      - Contigs are not expected to overlap much
      - Initial exonerate search is filtered for hits that are above a certain threshold

Steps from sequencing files to species trees II

- Contigs that pass this filter are arranged in order along the alignment
- All contigs that pass the previous steps are concatenated into a “supercontig” and the exonerate search is repeated

5. Search for paralogs — if SPAdes assembler generates multiple contigs that contain coding sequences representing 75% of the length of the reference protein, HybPiper will print a warning for that gene
6. Recovering of the individual sequences
7. Statistics of the recovery
8. Cleanup of temporal files (especially SPAdes produces huge amount of data unneeded for further processing)

6. Statistics of sequence lengths in all samples and more information about recovered contigs
7. Creation of heatmaps of sequence recoveries
8. Summary of warnings about paralogs
9. Retrieve of sequences of exons, introns and supercontigs for all samples

Steps from sequencing files to species trees III

10. Alignment of all contigs (e.g. by MAFFT; or MUSCLE, Clustal, ... e.g. using R and packages ape and/or ips)
    - All alignments must be trimmed — columns/rows with too much missing data (e.g. beginning and end of the alignment) must be removed (e.g. using R and package ape)
    - It is also useful to create simple NJ tree graphical check of alignment (e.g. using R and package ape)
11. Sorting of alignments, statistics of their length and quality, discarding of poor (too short, too few individuals, too few variable positions, ...) alignments
12. Reconstruction of gene trees from all aligned contigs (e.g. using IQ-TREE, or ExaML, MrBayes, PhyML, RAxML-NG, ...)
13. Post-processing of gene trees
    - Identification, inspection and possible removal of gene trees with significantly different topology (e.g. by R and packages ape and kdetrees; TreeShrink, etc.)

Steps from sequencing files to species trees IV

- Comparison of gene trees (e.g. heatmaps and PCoA by R and packages ade4, ape, distory, phytools, etc.)
- Comparison of (several) (species) trees (e.g.by R and packages ape or phytools)

14. Construction of species trees (e.g. by ASTRAL-III, ASTER* or ASTRID-2)
    - Comparison of species tree and gene trees (e.g. by phyparts and MJPythonNotebooks)
15. Phylogenetic networks (e.g. by PhyloNet)
16. And more...

Note...

- This general scheme can be significantly altered...
- There are plenty of technical as well as biological problems (HGT, ILS, ...) and new software keep being developed...
- Much more analysis possible...

List of software used during the course I

- ASTRAL (see lesson by TF) — species trees from gene trees
- BASH 4 or later and GNU core utils (“Linux command line”)
- BBMap — deduplication of FASTQ
- BLAST+ (used by HybPiper)
- BWA (used by HybPiper)
- Dendroscope — visualize outputs of PhyloNet
- Exonerate (used by HybPiper)
- GNU Parallel (used by HybPiper and in BASH scripts)
- HybPiper — recovering genes from targeted sequence capture data
- IQ-TREE — gene trees
- MAFFT — alignment

List of software used during the course II

- MJPythonNotebooks (used by phyparts)
- PhyloNet — phylogenetic networks
- phyparts — comparison of species tree vs. gene trees
- Python 3.7 or later and Biopython 1.80 or later (used by HybPiper)
- R 4.0 or later and packages ade4, adegenet, ape, corrplot, distory, ggplot2, gplots, ips, kdetrees, pegas, phangorn, phytools and scales
  - Used for alignment of contigs, post-processing of alignments, post-processing and comparison of gene trees, etc.
- Samtools (used by HybPiper)
- SPAdes (used by HybPiper)
- TreeShrink — detection of outlier long branches in collections of phylogenetic trees
- Trimmomatic — trimming of FASTQ

CESNET and MetaCentrum I

- CESNET (česky) is organization of Czech universities, Academy of Science and other organizations taking care about Czech backbone Internet, one of world leading institutions of this type
- CESNET provides various services (česky)
  - Massive computations — MetaCentrum (česky) — this we need to process our HybSeq data
  - Large data storage (česky) — this we can use to store our HybSeq data
  - And much more (česky) — note especially FileSender (česky) and ownCloud (česky)...
  - See also my course `https://soubory.trapa.cz/linuxcourse/linux_bash_metacentrum_course.pdf` (chapter "MetaCentrum")
- Information about data storage `https://du.cesnet.cz/en/start` (česky) contains detailed usage instructions
- Information about MetaCentrum `https://www.metacentrum.cz/en/` (česky)
- Most of practical information for users are at `https://docs.metacentrum.cz/`
  - Old wiki is deprecated

CESNET and MetaCentrum II

- To start work see at least access, computing, work with data and some tutorial
- Of course, good knowledge of work in Linux command line (BASH) is needed...

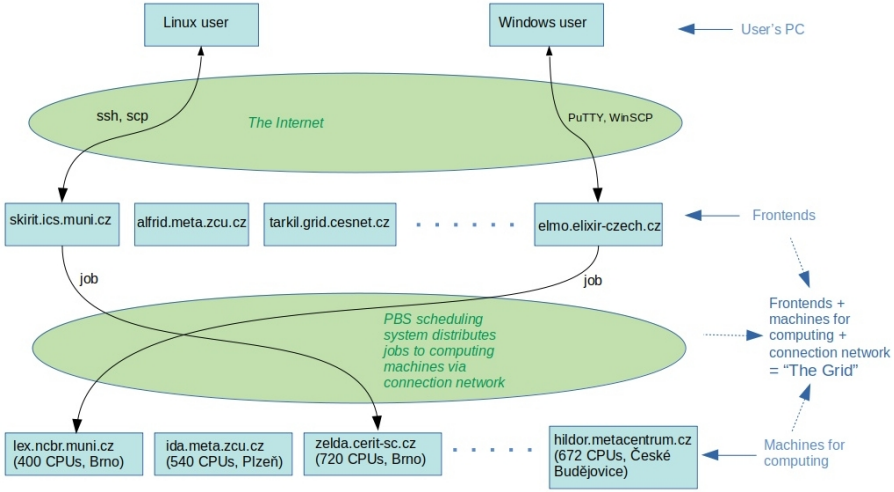
MetaCentrum vs. other grids and clusters...

- I show processing on MetaCentrum Czech National Grid Infrastructure, as it is readily available, well maintained and contains all needed applications, but it's possible to use any computing cluster in similar way (the scripts are easily modifiable)
- For other clusters than Czech MetaCentrum, user must in the scripts edit at least commands to load modules and handling with scratch directory
- MetaCentrum is distributed — it has multiple front end and storages, which might be confusing (be careful with paths in scripts)

MetaCentrum

- Find all needed information at https://docs.metacentrum.cz/
- Current state and usage as available at https://metavo.metacentrum.cz/
- Manage your user account at http://metavo.metacentrum.cz/en/myaccount/ (česky)
- Personal view on actual resources and running tasks is at https://metavo.metacentrum.cz/pbsmon2/person
- Listing of available applications https://docs.metacentrum.cz/software/search-soft/
- It has several front ends where users log, various storages, and thousands of computers (nodes) doing the calculations — they are not accessed directly to run task
  - Distributed nature and number of front ends and storages may be confusing for beginners
- Most of computers are running Debian GNU/Linux

Basic workflow



Launching of tasks

- https://docs.metacentrum.cz/computing/run-basic-job/
- Personal view https://metavo.metacentrum.cz/pbsmon2/person has nice overview of available resources and tasks and allows comfortable construction of submission command

```
# Task in metacentrum.sh script will run up to 4 days (96 h),
# require one physical computer with 8 CPU threads, 24 GB of
# RAM, 10 GB of disk space and we get all information mails
qsub -l walltime=96:0:0 -l \
  select=1:ncpus=8:mem=24gb:scratch_local=10gb -m abe metacentrum.sh
# Check how the task is running (above web) and
qstat -u $USER # Information about $USER's jobs
qstat 123456789 # The task ID is available from qstat
qstat -f 123456789 # Print a lot of details
qdel 123456789 # Terminate scheduled or running task
```

Key MetaCentrum commands

- MetaCentrum is "just" normal Linux server — work as usually
- Command `module` loads/unloads selected application (`module add r`)
- Tasks (BASH scripts) are submitted for computing by `qsub` — the script must copy the data into `$SCRATCHDIR` and do all calculations there
  - It has plenty of options how to specify requirements (see help)
- Queued and running jobs can be seen by `qstat -u $USER` (`qstat` has much more options) and any job can be terminated by `qdel 123456789` (number from `qstat`)

```
module avail XXX/ # List available modules for XXX
module add <TAB><TAB> # Load some module (or use 'module load XXX')
module list # List of currently loaded modules
module rm XXX # Unload selected module
# All jobs for $USER ('-x' also shows tasks ended within last 24 h)
qstat -w -n -1 -u $USER -t -p -a @pbs-m1.metacentrum.cz -x
```

Running R tasks on MetaCentrum

- There are only some R packages, to get more create own package library and use it in scripts (see e.g. `.libPaths()` within `R`)
- Be careful about paths!
- In the `metacentrum.sh` script load R, e.g. `module add r/4.1.3-gcc-10.2.1-6xt26dl` and start there R script as usually `R CMD BATCH script.r`

1. Login to selected front node via SSH and load **R** module
2. Create somewhere new directory for R packages `mkdir rpkgs` (or use default `~/R/`)
3. Start R `R` and install **all** R packages needed for the task — install them e.g. into the `rpkgs` directory `install.packages(pkgs=…, …, lib="rpkgs")`
4. In the R script `*.r` load the packages from the `rpkgs` directory `library(package=…, lib.loc="/storage/…/rpkgs")` or so
5. Ensure all needed outputs are saved from the R script

Test data directory structure I

```
├── hybseq # BASH scripts to process the data
I   ├── bin # Scripts themselves
I   ├── ref # References for HybPiper
I   ├── rpackages # R library with installed R packages
└── hybseq_course_zingibers # Data
    ├── 1_data # Sequencing libraries
    I   ├── lib_01 # Sequencing library 1
    I   I   ├── 0_data # Raw FASTQ sequences
    I   I   ├── 1_trimmed # Trimmed FASTQ sequences
    I   I   ├── 2_dedup # Deduplicated FASTQ sequences - for HybPiper
    I   I   └── 3_qual_rep # FastQC quality reports
    I   └── lib_02 # Possible sequencing library 2
    I       ├── 0_data # Raw FASTQ sequences
    I       ├... # Same structure as lib_01...
    ...     ... # Next slide...
```

Test data directory structure II

```
... # Previous slide...
├── 2_seqs # Outputs of HybPiper
I   ├── Aframomum-alboviolaceum_S118_L001.dedup # Sample output dir
I   I   ├── Assembly_1 # Gene output directory
I   I   ├── ... # More gene output directories...
I   I   └── Assembly_16358 # Gene output directory
I   ├── ... # More samples...
I   └── Zingiber-officinale_S242_L001.dedup # Sample output dir
I       ├── Assembly_1 # Gene output directory
I       ├── ... # More gene output directories...
I       └── Assembly_16358 # Gene output directory...
I    # Plenty of reconstructed contigs, statistics, etc.
├── 3_aligned # Aligned contigs
I   ├── exons # Aligned exons
I   ├── introns # Aligned introns
I   └── supercontigs # Aligned supercontigs
... # Next slide...
```

Test data directory structure III

```
... # Previous slide...
├── 4_gene_trees # Reconstructed gene trees
I   ├── exons # Gene trees of exons
I   ├── introns # Gene trees of introns
I   └── supercontigs # Gene trees of supercontigs
I   # Possibly create some substructure for outputs of kdetrees,
I   # TreeShrink and another gene trees filtration... e.g.
I   # mkdir {exons,introns,supercontigs}_{0_trees,1_unfiltered,
I   #   2_pcoa_kdetrees,3_treeshrink}
I   # To keep everything sorted
└── 5_species_trees # Processing of the trees
    ├── # Outputs of various analysis like ASTRAL...
    ... # More downstream analysis...
```

- Of course, every user can figure different directory structure, but HybSeq produces a lot of data and plenty of software packages are used, so keep some logical structure...

Data download

- MetaCentrum storages have sometimes too limited quota for such a large data — see your quotas (česky), see details
- The pipeline produce a lot of data (especially HybPiper; gene trees can be also large) — ensure to have enough space to store everything
  - Especially **HybPiper produces a lot of files** — user may reach quota for number of files, not necessarily (only) amount of data

```
# Login to any MetaCentrum front end
ssh USER@skirit.metacentrum.cz # Or any other front end
# Download the data
wget https://botany.natur.cuni.cz/zeisek/hybseq_course_zingibers_0_data.zip
# Unpack it
unzip hybseq_course_zingibers_0_data.zip
```

- The archive contains input raw data and reference
- Further steps are in other archives (see later)

Scripts and other resources to process the data

- The archive `hybseq_course_zingibers_0_data.zip` contains directory `hybseq_course_zingibers` with...
  - Raw data `*.fastq.gz` in `1_data/lib_01/0_data/`
  - Reference for HybPiper `curcuma_HybSeqProbes_first958_concat.fasta` in `ref/`

Do not blindly copy-paste commands...

Commands show typical way how to proceed, but should not be using without understanding, and other ways how to work are possible... Command and submission scripts also **do require** edits prior launching them.

Getting HybSeq scripts

- See https://github.com/V-Z/hybseq-scripts
- Install the scripts as follows:

```
# Ensure to be in home folder
cd
# Clone repository with scripts into "hybseq" directory
git clone https://github.com/V-Z/hybseq-scripts.git hybseq
# Install all needed R packages
cd hybseq/ # Go to hybseq directory
ls -lha # See content...
module add r/4.1.3-gcc-10.2.1-6xt26dl # Load R module
R # Start R and install needed packages (next slide)
```

Install needed R packages

```
getwd() # Ensure to be in ~/hybseq/ And if not go there by
setwd("~/hybseq/") # Set path into HybSeq directory
dir() # Ensure to see directory "rpackages"
# Package Matrix (required by ips) requires R 4.4 and newer while we have
# 4.1.3 - we have to install its dependency lattice and older version of
# Matrix manually
install.packages(pkgs="lattice", lib="rpackages",
  repos="https://mirrors.nic.cz/R/", dependencies="Imports")
install.packages(pkgs="https://cran.r-project.org/src/contrib/Archive/
  Matrix/Matrix_1.6-5.tar.gz", lib="rpackages", repos=NULL,
  dependencies="Imports")
# Install needed packages in R
install.packages(pkgs=c("ape", "codetools", "cpp11", "farver",
  "ips", "RcppArmadillo", "scales"), lib="rpackages",
  repos="https://mirrors.nic.cz/R/", dependencies="Imports")
dir("rpackages/") # Ensure all packages are correctly installed
```

Trimming and deduplication

- Raw demultiplexed FASTQ sequences must be trimmed (sequencing adaptors removed, ...) and should be deduplicated (removal of artificial duplicates to get correct statistics of coverage)
- There are plenty of software packages available, Trimmomatic use to be used for trimming and e.g. BBMap for deduplication
- Usually, libraries are processed as they are delivered from sequencing company
- Quality of all FASTQ files should be checked by e.g. FastQC
- It is practical to obtain simple statistics — number of sequences in original files, after trimming and after deduplication
- Low quality files should be discarded...
- Everything can be easily coded into simple BASH script processing all files (see following slides)

Scripts to trim and deduplicate FASTQ sequences

- Script `~/hybseq/bin/hybseq_1_prep_1_qsub.sh`
  - Submits via `qsub` script `~/bin/hybseq_1_prep_2_run.sh` FASTQ data for trimming, deduplication and quality checks
  - Variables `WORKDIR` (location of HybSeq scripts) and `DATADIR` (data to process) must point to correct existing location — script contains settings for running following script
  - If changing parameters for `hybseq_1_prep_2_run.sh`, parameters for `qsub` must be changed here accordingly
  - The script is started multiple times to process all sequencing libraries
- Script `~/hybseq/bin/hybseq_1_prep_2_run.sh`
  - See `./hybseq_1_prep_2_run.sh -h` for usage help
  - Script checks if all needed parameters and tools are available and in simple `for` loop trimms all sequences (one by one), deduplicates them, does quality checking, prints simple statistics, and prepares list of samples for HybPiper
- Output is prepared as input for HybPiper itself

Submission of hybseq_1_prep_1_qsub.sh

- Pre-processing data for HybPiper
- `~/hybseq/bin/hybseq_1_prep_1_qsub.sh` must be edited before submission via `qsub` — change `WORKDIR` and `DATADIR`
- After it runs for a while (everything had been copyied to the computing node), it is possible to change `DATADIR` and submit processing of the second library

```
# After edition of WORKDIR and DATADIR, go to data directory...
cd ~/hybseq_course_zingibers/1_data/lib_01/ # ...and submit the job...
qsub -l walltime=24:0:0 -l \
  select=1:ncpus=4:mem=64gb:scratch_local=100gb -m abe \
  ~/hybseq/bin/hybseq_1_prep_1_qsub.sh
# Or similar command
# Monitor running $USER's tasks with details
qstat -w -n -1 -u $USER # Last column contains machine name
# See your processes running the machine (from above list)
ssh exec_host "ps ux" # Replace exec_host by hostname!
```

Trimm and deduplicate all FASTQ files

Tasks to pre-process FASTQ data for HybPiper

1. Inspect `~/hybseq/bin/hybseq_1_prep_1_qsub.sh` and `~/hybseq/bin/hybseq_1_prep_2_run.sh` and be sure to understand what the scripts do, including syntax used.
2. Edit declarations of variables `WORKDIR` and `DATADIR` in `hybseq_1_prep_1_qsub.sh` so that they point to correct locations.
3. Submit via `qsub hybseq_1_prep_1_qsub.sh` to process the data and monitor the jobs during processing.
4. Inspect outputs of `hybseq_1_prep_2_run.sh`, including statistics and FASTQ checks. What do they show?
5. Can you run the task on your computer (desktop or notebook) without `qsub`? If so, how?

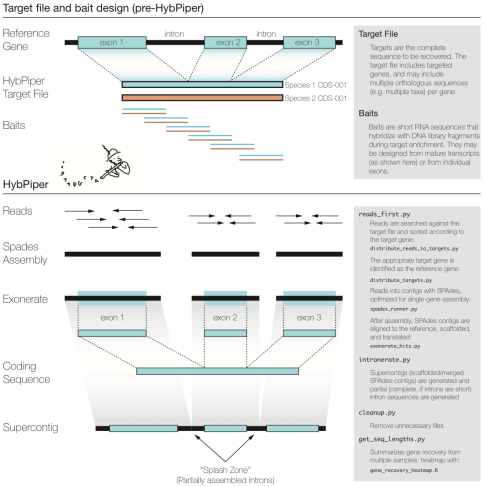
When the pre-processing job is done

- Wait for task to finish, or download and inspect test data
- Output directory `job_123456789.pbs-m1.metacentrum.cz` will be in `DATADIR` (`0_data`) — go there and move directories `1_trimmed`, `2_dedup` and `3_qual_rep` and log file `hybseq_prepare.log` to same level as `0_data`, e.g.
  `mv 1_trimmed 2_dedup 3_qual_rep hybseq_prepare.log ../../`
- Check log files `hybseq_prepare.log` and `hybseq_1_prep_1_qsub.sh.[eo]123456789`
- Directory `1_trimmed` contains statistics in `report_trimming.tsv`
- Directory `2_dedup` contains statistics in `report_filtering.tsv` and everything needed for HybSeq (see following chapter)
- See HTML files with quality reports in directory `3_qual_rep`

HybPiper

- See README and wiki documentation
- HybPiper was designed for targeted sequence capture, in which DNA sequencing libraries are enriched for gene regions of interest, especially for phylogenetics
- The HybPiper pipeline starts with high-throughput sequencing reads, and assigns them to target genes using BLASTx/DIAMOND or BWA. The reads are distributed to separate directories, where they are assembled separately using SPAdes. The main output is a FASTA file of the (in frame) CDS portion of the sample for each target region, and a separate file with the translated protein sequence.
- Includes commands to extract the intronic regions flanking each exon, and investigate putative paralogs
- End with contigs (one for each probe) ready to align

Overview of HybPiper



Requirements to run HybPiper

- See `https://github.com/mossmatters/HybPiper#readme` — HybPiper has plenty of requirements, the easiest is probably installation using `conda`
- It is possible to run HybPiper on your computer (with Linux or macOS), but it requires plenty of CPU and memory and creates huge output directories...
- `hybpiper assemble` is processing single sample — to process multiple (by `while` loop or by `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh`) there must be list of sample base names without suffixes like `*[._]R{1,2}[._]*f*q*` (here `samples_list.txt` created by `hybseq_1_prep_2_run.sh`)
- Reference bait FASTA file **must** have sequences named as `>Species_name-gene_id` (see help) (note order and dash in between)
- HybPiper is processing individual files with given baits FASTA sequences — batch processing must be scripted

Before running HybPiper

- Directory `~/hybseq/ref/` contains prepared reduced bait file `curcuma_HybSeqProbes_first958_concat.fasta` (it will be used to process test data)
- To run HybPiper we need the target reference file above and all required software, and BASH script to process all input files in batch — see `hybseq_2_hybpiper_1_submitter.sh` (submitting via `qsub` all input files), `hybseq_2_hybpiper_2_qsub.sh` (preparing individual job to run) and `hybseq_2_hybpiper_3_run.sh` (processing every input file — doing the job)
- Settings `WORKDIR`, `DATADIR`, `SAMPLES`, `BAITFILE` and `NCPU` **must be changed** in `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh`

Prepare to run HybPiper

Tasks before running HybPiper

1. Check `samples_list.txt` in `2_dedup` output directory of the test data after running `~/hybseq/bin/hybseq_1_prep_1_qsub.sh`, and check how it was created.
2. Compare `https://github.com/tomas-fer/HybPhyloMaker/blob/master/HybSeqSource/curcuma_HybSeqProbes_test.fa` (output of Geneious assembler) and `curcuma_HybSeqProbes_first958_concat.fasta` — see form required by HybPiper
3. Do we have everything needed to run HybPiper?

Understanding how presented scripts run HybPiper I

- Script `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh` goes to directory set by `DATADIR` variable and for every sample (pair of forward and reverse FASTQ files) listed in `samples_list.txt` (variable `SAMPLES`, created by `hybseq_1_prep_2_run.sh`) and submits via `qsub` individual task for each input sample (thy are processed independently in parallel)
- Script `~/hybseq/bin/hybseq_run_2_hybpiper_2_qsub.sh` drives submission and manipulation of processing of each individual file — it checks if all required data are provided, copy input files to `SCRATCHDIR`, loads required software modules, runs processing itself by `hybseq_2_hybpiper_3_run.sh` and copy results back
- Script `~/hybseq/bin/hybseq_2_hybpiper_3_run.sh` runs all HybPiper steps for individual samples (as submitted by previous scripts)

Understanding how presented scripts run HybPiper II

- Outputs of `~/hybseq/bin/hybseq_2_hybpiper_3_run.sh` should be checked and moved into special directory (`~/hybseq_course_2023_zingibers/2_seqs`) where new list of samples must be created
- Script `~/hybseq/bin/hybseq_run_3_hybpiper_postprocess_1_qsub.sh` can be then used to run `~/hybseq/bin/hybseq_3_hybpiper_postprocess_2_run.sh` which uses HybPiper to retrieve sequences of exons, introns and supercontigs; prints statistics, creates heatmaps, and summaries paralogs warnings — retrieved sequences can then be aligned (see further)

Submitting jobs to run HybPiper

- To retrieve probe sequences from every input FASTQ file
- `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh` must be edited before running it — change `WORKDIR`, `DATADIR`, `SAMPLES`, `BAITFILE` and `NCPU` according to your needs
- After submission of all input files is done, it is possible to change `DATADIR` and submit processing of the second library
- It will start job for every sample, so that output of `qstat` will notably prolong...

```
# After edition of WORKDIR, DATADIR, SAMPLES, BAITFILE and NCPU, go to
# DATADIR AND run simply (There are no further parameters on command line)
cd ~/hybseq_course_zingibers/1_data/lib_01/2_dedup/ # Go to DATADIR
~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh # Submit all samples
# Monitor running $USER's tasks with details
qstat -w -n -1 -u $USER # Last column contains machine name
# See your processes running the machine (from above list)
ssh exec_host "ps ux" # Replace exec_host by hostname!
```

Running HybPiper

Run HybPiper

1. Inspect scripts `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh`, `~/hybseq/bin/hybseq_2_hybpiper_2_qsub.sh` and `~/hybseq/bin/hybseq_2_hybpiper_3_run.sh` and be sure to understand what they are doing, including syntax used.
2. Check https://github.com/mossmatters/HybPiper/wiki/Full-pipeline-parameters#10-hybpiper-assemble
3. Edit declarations of variables `WORKDIR`, `DATADIR`, `SAMPLES` and `BAITFILE` in `hybseq_2_hybpiper_1_submitter.sh` so that they point to correct locations.
4. Run in directory `2_dedup` script processing all samples, simply `~/hybseq/bin/hybseq_2_hybpiper_1_submitter.sh`

Other tasks with HybPiper

For interested students

HybPiper tasks for advanced users

1. Think how to process all samples on single computer. Use `while` BASH loop and feed it by `samples_data.txt`. How would such script look like? What had to be changed?
2. Check requirements of software used by HybPiper (BWA, SPAdes, ...) and think if changing of number of CPU threads and memory would significantly speed up processing of individual files.
3. See `man qsub` and think if there is another option how to pass options from `hybseq_2_hybpiper_1_submitter.sh` for individual `qsub` commands (apart of usage of exported variables as is used now).
4. Think about HybPiper parameters in `hybseq_2_hybpiper_3_run.sh`.

After running HybPiper for every sample...

- Previous step processed by HybPiper every single sample individually and independently
- HybPiper postprocessing will retrieve contigs for every exon, intron and supercontig containing all individuals where particular genetic region was found; and do some statistics
- Script `~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh` is submitted via `qsub` and it runs `~/hybseq/bin/hybseq_3_hybpiper_postprocess_2_run.sh` which uses HybPiper to obtain table of contig lengths, some statistics, heatmaps, information about paralogs, and retrieves from individual directories contigs to be aligned
- New list of samples for all samples (all libraries) must be prepared (next slide)
- All outputs (a lot of files) are in same directory where HybPiper outputs are (typically `XXX/2_seqs/`)

Sorting data after running HybPiper for all samples

```
# Ensure You are in 2_dedup
cd ~/hybseq_course_zingibers/1_data/lib_01/2_dedup/
# Move MetaCentrum logs into output directories
while read L; do mv HybPiper."$L".[eo]* "$L"/; done < samples_list.txt
# Move from ~/hybseq_course_zingibers/1_data/lib_01/2_dedup all outputs of
# HybPiper to hybseq_course_zingibers/2_seqs/ for next steps
mkdir ../../../2_seqs # Create directory for HybPiper outputs
mv *.dedup ../../../2_seqs/ # Move all HybPiper output directories there
cd ../../../2_seqs/ # Go to that directory
# NOTE: The mkdir command is needed only once and directory 2_seqs can
# contain results from multiple sequencing libraries
```

- In `~/hybseq_course_zingibers/1_data/lib_#/2_dedup/` can be multiple libraries — all HybPiper outputs should be moved into `~/hybseq_course_zingibers/2_seqs/` where HybPiper postprocessing will retrieve all sequences and statistics

HybPiper postprocessing — retrieving of contigs and statistics

- Code below is exemplary and requires edits, as well as script `~/hybseq/bin/hybseq_run_3_hybpiper_postprocess_1_qsub.sh` — change `WORKDIR`, `BAITFILE`, `DATADIR` and `SAMPLES`
- This process can be repeated every time new sequencing library is added
- All outputs will be directly in `DATADIR` (`2_seqs`)

```
# Create in ~/hybseq_course_zingiberss/2_seqs/ new samples_list.txt
find . -maxdepth 1 -type d | sed 's/^\.\///' | sort | tail -n+2 > \
  samples_list.txt
# Edit WORKDIR, BAITFILE and DATADIR in
# ~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh
# When ready, submit task to post-process HybPiper results
qsub -l walltime=12:0:0 -l select=1:ncpus=1:mem=8gb:scratch_local=100gb \
   -m abe ~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh
# Note that for large data (~1000 probes and hundreds of samples) the
# postprocessing can take long time and longer walltime can be needed
```

Post-processing HybPiper outputs and retrieving contig sequences I

Tasks to post-process HybPiper outputs I

1. Inspect `~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh` and `~/hybseq/bin/hybseq_3_hybpiper_postprocess_2_run.sh` and be sure to understand what the scripts do, including syntax used.
2. Edit declaration of variables `WORKDIR`, `BAITFILE` and `DATADIR` in `~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh` so that they point to correct locations.
3. Submit via `qsub` `~/hybseq/bin/hybseq_3_hybpiper_postprocess_1_qsub.sh` to post-process all HybPiper outputs.
4. Inspect outputs, including statistics, heatmaps and log. What do they show?

Post-processing HybPiper outputs and retrieving contig sequences II

For interested students

- Wait for tasks to finish, or download and inspect test data

Tasks to post-process HybPiper outputs II

1. Which part does take the longest time on this task? Does this task take plenty of resources (CPU, memory)?
2. Check parameters of `hybpiper stats`, `recovery_heatmap`, `retrieve_sequences` and `paralog_retriever` at `https://github.com/mossmatters/HybPiper/wiki/Full-pipeline-parameters`
3. Can you run the task on your computer (desktop in office or notebook) without `qsub`? If so, how? Can you run the task directly on MetaCentrum front end?

Alignment of all contigs I

- All sequences retrieved in the previous step with HybPiper must be aligned (by any aligner)
- Alignments must be post-processed
  - Rows (individuals) and/or colums (positions within alignment) with more than ∼10–50% of missing data must be removed (e.g. beginning and the end of the alignment)
  - Too short alignments or alignments with too few variable sites or too few individuals could be removed (their usefulness is questionable)
  - Exact thresholds are to be discussed (according to purpose, phylogenetic scale etc.)...
- Script `~/hybseq/bin/hybseq_4_alignment_1_submitter.sh` goes to directory set by `DATADIR` variable and for every contig (all `*.fasta` or `*.FNA` files) in that directory (retrieved by `hybseq_3_hybpiper_postprocess_2_run.sh`) submits via `qsub` individual alignment task

Alignment of all contigs II

- Script `~/hybseq/bin/hybseq_run_4_alignment_2_qsub.sh` drives submission and manipulation of processing of each individual file — it checks if all required data are provided, copy input files to `SCRATCHDIR`, loads required software modules, runs processing itself by `hybseq_4_alignment_3_run.r` `R` script and copy results back
- `R` script `~/hybseq/bin/hybseq_4_alignment_3_run.r` aligns every input contig with MAFFT, trims the alignment, creates NJ tree (NWK and PNG), creates image of alignment and reports alignment details
- Script `~/hybseq/bin/hybseq_4_alignment_4_postprocess.sh` is short, can be runned on the front end, it will sort outputs into subdirectories for exons, intron and supercontigs, create statistics of alignments and lists of NJ gene trees

Submitting alignment jobs

- To align all retrieved contigs (sequences)
- `~/hybseq/bin/hybseq_4_alignment_1_submitter.sh` must be edited before running it — change `WORKDIR` and `DATADIR`
- It will start job for every sample, so that output of `qstat` will be very long (3 jobs — respective exon, intron and supercontig — for every probe)...
- For small contigs, jobs are very fast

```
# After edition of WORKDIR and DATADIR run simply
# NOTE: There are no further parameters on command line
~/hybseq/bin/hybseq_4_alignment_1_submitter.sh
# Monitor running $USER's tasks with details
qstat -w -n -1 -u $USER # Last column contains machine name
# Something went wrong? Cancel all running or queued tasks by
qdel $(qstat -u $USER | grep -o "^[0-9]\+" | tr "\n" " ")
```

Running alignments

Run alignments

1. Inspect scripts `~/hybseq/bin/hybseq_4_alignment_1_submitter.sh`, `~/hybseq/bin/hybseq_4_alignment_2_qsub.sh` and `~/hybseq/bin/hybseq_4_alignment_3_run.r` and be sure to understand what they are doing, including syntax used.
2. Edit declarations of variables `WORKDIR` and `DATADIR` in `hybseq_4_alignment_1_submitter.sh` so that they point to correct locations.
3. Process all `*.FNA` and `*.fasta` files.
4. Monitor progress of the jobs
5. Inspect outputs (including images) and log files.

Other tasks with alignments

For interested students

Alignment tasks for advanced users

1. Think how to process all samples on single computer. Use `find` to list all `*.FNA` and `*.fasta` files and pass them to GNU Parallel. How would such script look like? What had to be changed?
2. Check requirements of MAFFT and/or another aligners and think if changing of number of CPU threads and memory would significantly speed up processing of individual files.
3. In `hybseq_4_alignment_3_run.r` replace usage of MAFFT by another aligner like e.g. MUSCLE or Clustal.
4. Think about parameters for functions `deleteGaps()`, `del.rowgapsonly()` and `del.colgapsonly()`. How do they influence output (`*.aln.fasta` files)?

After all alignment jobs are done

- Results (alignments named `*.aln.fasta` and other files) are in newly created `aligned` directory created by `hybseq_4_alignment_1_submitter.sh` in `DATADIR`
- Other outputs are images with alignment checks (`*.aln.check.png` and `*.aln.png`), NJ trees (`*.nwk` and `*.tree.png`), saturation plots (`*.saturation.png`) and logs (`*.log` from `R` and `HybSeq.alignment.*.[eo]*` from `qsub`)
- Outputs should be sorted by `hybseq_4_alignment_4_postprocess.sh` — it requires requires two arguments: directory to process (`aligned`) and path to list of samples (i.e. `samples_list.txt` in `2_seqs`)

Sorting data after running alignments for all samples

- Outputs of alignments are in directory `aligned` which was created in the input directory, files with alignments itself are named `*.aln.fasta`
- It should be sorted by `hybseq_4_alignment_4_postprocess.sh`
- All outputs should be then moved to dedicated directory (cf. slide 21)
- Code below is exemplary and requires edits

```
# Move qsub logs to the 'aligned' directory - go to directory XXX/2_seqs
mv HybSeq.alignment.* aligned/
# Everything should be moved from XXX/2_seqs/aligned to XXX/3_aligned
mv aligned ../3_aligned && cd ../
# Post-process (sort into subdirectories and get simple statistics)
# all alignments - provide path to directory with aligned files
~/hybseq/bin/hybseq_4_alignment_4_postprocess.sh -p 3_aligned \
  -s 2_seqs/samples_list.txt | tee hybseq_align_postprocess.log
mv hybseq_align_postprocess.log 3_aligned/ && cd 3_aligned/
```

Post-processing of alignments

Tasks to post-process alignments

1. Inspect `~/hybseq/bin/hybseq_4_alignment_4_postprocess.sh` and be sure to understand what the script does, including syntax used.
2. Run `~/hybseq/bin/hybseq_4_alignment_4_postprocess.sh` with correct paths to post-process all aligned outputs (previous slide).
3. Inspect outputs, including statistics, images and logs. Open in spreadsheet (e.g. LibreOffice Calc) the `*.tsv` tables. What do they show?

- Wait for tasks to finish, or download and inspect test data

Gene trees from all alignments I

- Gene trees must be computed from all aligned sequences
- Gene trees must be post-processed
  - Trees should be sorted into subdirectories for exons, introns and supercontigs and lists of gene trees created
  - Trees with significantly different topology must be identified and inspected (and possibly removed) — this will be later done in `R`
  - Long branches within trees must be identified and respective trees inspected — artificial long branches can be discarded from given trees (e.g. by TreeShrink)
- Script `~/hybseq/bin/hybseq_5_gene_trees_1_submitter.sh` goes to directory set by `DATADIR` and for every aligned contig (all `*.aln.fasta` files) in that directory (created in previous step) submits via `qsub` individual task to reconstruct gene tree

Gene trees from all alignments II

- Script `~/hybseq/bin/hybseq_5_gene_trees_2_qsub.sh` drives submission and manipulation of processing of each individual file — it checks if all required data are provided, copy input files to `SCRATCHDIR`, loads required software modules, runs processing itself by `hybseq_5_gene_trees_3_run.sh` and copy results back
- Script `~/hybseq/bin/hybseq_5_gene_trees_3_run.sh` computes gene tree for every input file with IQ-TREE 2 (see how easily it can be converted for another tree builder, e.g. RAxML-NG)
- Script `~/hybseq/bin/hybseq_5_gene_trees_4_postprocess.sh` is short, can be runned on the front end, it will sort outputs into subdirectories for exons, intron and supercontigs and create lists of maximum likelihood and consensus (after bootstrapping) gene trees

Submitting gene trees jobs

- To reconstruct gene trees from every aligned and trimmed sequence
- `~/hybseq/bin/hybseq_5_gene_trees_1_submitter.sh` must be edited before running it — change `WORKDIR` and `DATADIR`
- It will start job for every sample, so that output of `qstat` will be very long (3 jobs — respective exon, intron and supercontig — for every of ∼1000 probes)...
- Larger alignments can take long time to compute — set `walltime` in `hybseq_5_gene_trees_1_submitter.sh` accordingly

```
# After edition of WORKDIR and DATADIR run simply
# NOTE: There are no further parameters on command line
~/hybseq/bin/hybseq_5_gene_trees_1_submitter.sh
# Monitor running $USER's tasks with details
qstat -w -n -1 -u $USER # Last column contains machine name
# See your processes running the machine (from above list)
ssh exec_host "ps ux" # Replace exec_host by hostname!
```

Running gene trees

Run gene trees

1. Inspect scripts `~/hybseq/bin/hybseq_5_gene_trees_1_submitter.sh`, `~/hybseq/bin/hybseq_5_gene_trees_2_qsub.sh` and `~/hybseq/bin/hybseq_5_gene_trees_3_run.sh` and be sure to understand what they are doing, including syntax used.
2. Edit declarations of variables `WORKDIR` and `DATADIR` in `hybseq_run_5_gene_trees_1_submitter.sh` so that they point to correct locations.
3. Run it to process all `*.aln.fasta` files.
4. Monitor progress of the jobs
5. Inspect outputs and log files.

Other tasks with gene trees

For interested students

Gene trees tasks for advanced users

1. See IQ-TREE help, check its parameters in `hybseq_5_gene_trees_3_run.sh` and think about possible changes to fit better your needs.
2. Think how to process all samples on single computer. Use `find` to list all `*.aln.fasta` files and pass them to GNU Parallel. How would such script look like? What had to be changed?
3. Check requirements of IQ-TREE and/or another tree builder and think if changing of number of CPU threads and memory would significantly speed up processing of individual files.
4. Replace usage of IQ-TREE in `hybseq_5_gene_trees_3_run.sh` by another tree builder like e.g. ExaML, MrBayes, PhyML or RAxML-NG.

Quick visualization of gene trees I

- Individual gene trees can be quickly visualized by simple **R** script processing in `for` loop each gene tree input file
- Adjust code below and save it as `mltreeplot.r` in directory with gene trees

```
library(ape, lib.loc="~/hybseq/rpackages")
# [1] file.aln.fasta.treefile / file.aln.fasta.raxml.support
fnames <- commandArgs(TRUE)
tr <- read.tree(file=fnames[1])
png(filename=fnames[2], width=1800, height=1800, units="px", bg="white")
  plot.phylo(x=tr, type="unrooted", edge.color="blue", edge.width=3,
    cex=1.25)
  title("Maximum-likelihood phylogenetic tree")
  nodelabels(text=tr$node.label, frame="none", col="red")
  edgelabels(text=round(x=tr$edge.length, digits=3), frame="none",
    col="brown", cex=0.75)
  add.scale.bar()
  dev.off()
```

Quick visualization of gene trees II

- Run following `for` loop to process `R` script from previous slide to get quick visualization of individual gene trees
- Of course, this is very exemplary and especially graphical parameters must be highly adjusted according size of phylogeny etc.

```
module add r/4.1.3-gcc-10.2.1-6xt26dl # Load R. Ensure 'ape' is installed.
# IQ-TREE
for L in *.aln.fasta.treefile; do echo "${L}"; R CMD BATCH --no-save \
  --no-restore "--args ${L} ${L%.aln.fasta.treefile}.mltree.png" \
  mltreeplot.r; done
# RAxML-NG
for L in *.aln.fasta.raxml.support; do echo "${L}"; R CMD BATCH --no-save \
  --no-restore "--args ${L} ${L%.aln.fasta.raxml.support}.mltree.png" \
  mltreeplot.r; done
# Note > " < around arguments to pass to R
```

After all gene trees jobs are done

- Results are in newly created `trees` directory created by `hybseq_5_gene_trees_1_submitter.sh` in `DATADIR`
  - Records of what IQ-TREE did (`*.log`) and `HybSeq*` from `qsub`
  - Results are in `*.iqtree` (IQ-TREE report), maximum likelihood tree is in `*.treefile` and likelihood distances in `*.mldist`
  - Ultrafast bootstrap approximation results contain split support values in `*.splits.nex`, consensus tree in `*.contree`, bootstrap trees `*.ufboot` and likelihood mapping plots in `*.ckp.gz`; and more...
- Outputs should be sorted by `hybseq_5_gene_trees_4_postprocess.sh` — it requires only path to the `trees` directory
- Outputs of gene trees are in directory `trees` which was created in the input directory

Sorting data after running gene trees for all samples

- Results should be sorted by `hybseq_5_gene_trees_4_postprocess.sh`
- All outputs should be then moved to dedicated directory (cf. slide 21)
- Code below is exemplary and requires edits

```
# Move qsub logs to the 'trees' directory
cd ~/hybseq_course_zingibers/3_aligned/ && mv HybSeq.genetree.* trees/
# Everything should be moved from XXX/2_seqs/aligned to XXX/3_aligned
mv trees ../4_gene_trees && cd ../
# Post-process (sort into subdirectories and get lists of gene trees)
# all gene trees - provide path to directory with gene trees files
~/hybseq/bin/hybseq_5_gene_trees_4_postprocess.sh 4_gene_trees | tee \
  hybseq_gene_trees_postprocess.log
```

- Wait for tasks to finish, or download and inspect test data

Post-processing of gene trees

Tasks to post-process gene trees

1. Inspect `~/hybseq/bin/hybseq_5_gene_trees_4_postprocess.sh` and be sure to understand what the script does, including syntax used.
2. Run `~/hybseq/bin/hybseq_5_gene_trees_4_postprocess.sh` with correct path to post-process all gene trees outputs.
3. Inspect outputs of, including logs. What do they show?

- Now we have gene trees for all alignments — we can directly construct species tree (using e.g. ASTRAL-III, ASTER* or ASTRID-2 from lists of trees `trees_ml_*.nwk`), or inspect and compare topologies of gene trees and possibly discard outliers

```
# Some SW like ASTRAL dislike tree names on the beginning of lines
# - discard them e.g. by (i.e. keeping only tree topology):
sed -i 's/^[[:graph:]]\+ //' trees_*.nwk
```

Seeing trees in forest

- Comparison of gene trees start with identifying trees with significantly different topology
- There are several distance matrices allowing compare topological differences among trees (and subsequently plot heatmap, PCoA, etc.)

How to recognize artifact and real biologial feature?

- Without good reference genome it is hard to tell if long branches, weird topologies, etc. are some artifacts or biological reality...
- Problems commonly start with low-quality DNA in lab and subsequent high number of missing data
- Statistically, most of "weird" gene trees topologies are rather from technical issues, so that most of people filter them out...
- Results can vary according to strictness with trimming raw FASTQ, sensitivity of various HybPiper settings, settings of aligner and tree builder...

Install needed R packages

- The tasks will be done in R (use e.g. RStudio or RKWard)
- Use your notebook or e.g. MetaCentrum OnDemand RStudio or Jupyter notebook
- Following R code is available at `https://github.com/V-Z/hybseq-course/blob/master/trees_filtration.r` — **download** it and edit according to your needs

```r
# Install R packages
install.packages(pkgs=c("ape", "ade4", "distory", "gplots", "ggplot2",
  "phangorn", "phytools"), repos="https://mirrors.nic.cz/R/",
  dependencies="Imports")
# Install kdetrees package (removed from CRAN)
# Ensure package 'devtools' is installed
if( ! 'devtools' %in% installed.packages()) {install.packages('devtools')}
# Install 'kdetrees' from https://github.com/V-Z/kdetrees Git repository
devtools::install_github('V-Z/kdetrees')
```

Distances comparing trees I

Single number to compare each pair of complex topologies?

- To compare topology of trees, we need some apropriate distance matrix
- There is no general agreement which is the best, all have issues...
- If the distance matrix is not Euclidean, we run into another issues...

- Download e.g. `trees_ml_exons.nwk` (or another final list of gene trees) and work in `R` in your notebook
  - Comparing plenty of individual gene trees, finding different topologies, construction of consensual species tree topology
  - Robinsons-Foulds distance in `phytools::multiRF`
    - The index adds 1 for each difference between pair of trees
    - Well defined only for fully bifurcating trees — if not fulfilled, some results might be misleading
    - Allow comparison of trees created by different methods

Distances comparing trees II

  - If the difference is very close to root, RF value can be large, even there are not much differences in the tree at all — `dist.multiPhylo` from package distory can be an alternative, although interpretation of that geodesic distance is sometimes not so straightforward as simple logic of RF
- Methods implemented in `ape::dist.topo` allow comparison of trees with polytomies (`method="PH85"`) or use of squared lengths of internal branches (`method="score"`)
- Final matrices are commonly not Euclidean — may be problematic for usage in methods like PCoA (it can show misleading results)
  - Test it with `ade4::is.euclid`, can be scaled (forced to became Euclidean) by functions like `quasieuclid` or `cailliez` in `ade4` — carefully, it can damage meaning of the data
  - We get matrix of pairwise differences among trees (from multiple genes), we need display and analyze it
- Set of tools for identifying discordant phylogenetic trees are e.g. in package kdetrees
- Filtered trees (with removed outlying topologies) are input for further species tree reconstruction method

Loading trees into R

```
# Load libraries
library(ape)
library(ade4)
library(distory)
library(gplots)
library(ggplot2)
library(kdetrees)
library(phangorn)
library(phytools)
# Set working directory
setwd("~/dokumenty/vyuka/hybseq/") # For example
# Load the list of trees
trees <- read.tree(file="trees_ml_exons.nwk")
trees # See it
print(trees, details=TRUE)
```

Heatmap of topological distances

- There are several heatmap functions, try also at least `heatmap`
- Edit settings to fit your needs and preferences

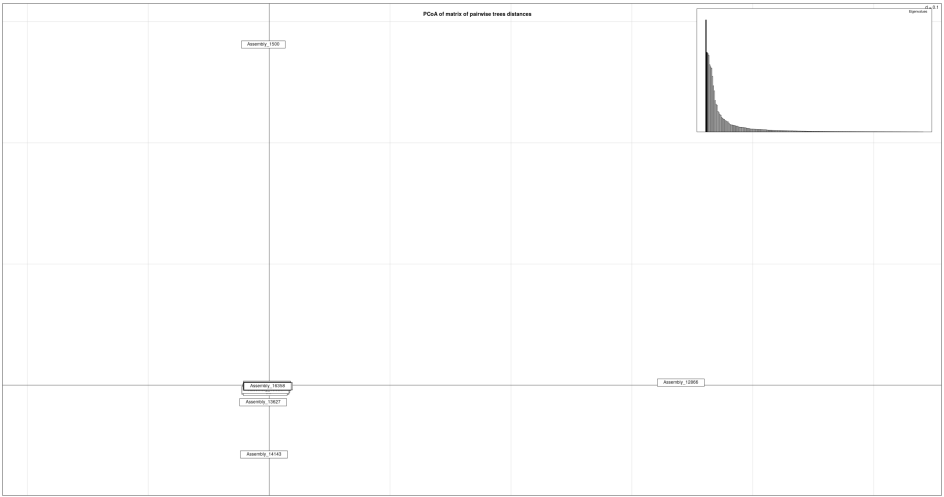
```
# Compute distance of topological similarities
# Set number of cores according to your computer
trees.d <- dist.topo(x=trees, method="score", mc.cores=4)
# Plot the heatmap (package gplots)
png(filename="trees_dist.png", width=10000, height=10000)
  heatmap.2(x=as.matrix(trees.d), Rowv=FALSE, Colv="Rowv",
    dendrogram="none", symm=TRUE, scale="none", na.rm=TRUE,
    revC=FALSE, col=rainbow(15), cellnote=as.matrix(trees.d),
    notecex=1, notecol="white", trace="none",
    labRow=rownames(as.matrix(trees.d)), labCol=colnames
    (as.matrix(trees.d)), key=FALSE, main="Correlation
    matrix of topographical distances")
  dev.off() # Saves the image
```

PCoA of topological distances I

- Requires Euclidean distance matrix (`is.euclid()`)
- Non-Euclidean matrices can be forced to became Euclidean by e.g. `quasieuclid()` or `cailliez()`
- There are plenty of options how to display it

```
# Test if the distance matrix is Euclidean
is.euclid(distmat=as.dist(trees.d), plot=TRUE, tol=1e-05)
# PCoA
trees.pcoa <- dudi.pco(d=trees.d, scannf=FALSE, nf=5)
trees.pcoa
# Plot PCoA
s.label(dfxy=trees.pcoa$li)
s.kde2d(dfxy=trees.pcoa$li, cpoint=0, add.plot=TRUE)
add.scatter.eig(trees.pcoa[["eig"]], 3,1,2, posi="topright")
title("PCoA of matrix of pairwise trees distances")
```

PCoA of topological distances II



Filtration of PCoA

```
trees # See original trees
# Remove trees identified in the PCoA plot
trees[c("Assembly_12866", "Assembly_14143", "Assembly_1500")] <- NULL
trees # See new object
# Possibly remove trees with too few tips
print(trees, details=TRUE)
trees[c(1, 2, 3, 4)] <- NULL
# Possibly remove rare tips
trees <- lapply(X=trees, FUN=drop.tip, tip=c("Amomum-sp7_S308_L001"))
class(trees) <- "multiPhylo" # Use after usage of lapply to multiPhylo
```

- Now you can repeat recalculation of distance matrix and PCoA and possibly remove more trees... — or use another method like kdetrees (next slide) etc.
- Calculation of distance matrix for large tree set can be very time demanding...
- See more details in https://soubory.trapa.cz/rcourse/, chapter “Trees” and subchapters “Seeing trees in forest” and “Comparisons”

Kdetrees

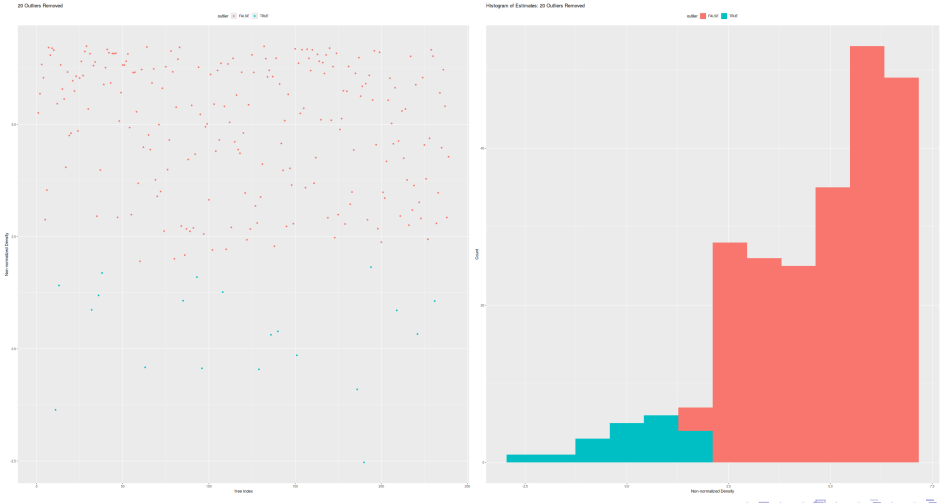
- Finds discordant phylogenetic trees
- Produces relative scores — high are relatively similar to each other, low dissimilar (discordant with the others)
- Produces scores, list of passing/discarded trees and graphical outputs
- In `kdetrees()`, value of `k` is responsible for threshold for removal of outliers — play with it

```
# Run kdetrees to detect outliers - play with k
?kdetrees # See options for kdetrees
trees.kde <- kdetrees(trees=trees, k=0.9 distance="dissimilarity",
  topo.only=FALSE, greedy=TRUE)
# See text results with list of outlying trees
trees.kde
```

Outputs of kdetrees

```
# See graphical results
plot(x=trees.kde)
hist(x=trees.kde)
# See removed trees
plot.multiPhylo(trees.kde[["outliers"]])
# Save removed trees
write.tree(phy=trees.kde[["outliers"]], file="trees_outliers.nwk")
# Save kdetrees report
write.table(x=as.data.frame(x=trees.kde), file="trees_scores.tsv",
  quote=FALSE, sep="\t")
# Extract passing trees
trees.good <- trees[names(trees) %in% names(trees.kde[["outliers"]])
  == FALSE]
trees.good
# Save passing trees
write.tree(phy=trees.good, file="trees_good.nwk")
```

Outputs of kdetrees — graphs



TreeShrink

- Algorithm for detecting abnormally long branches in one or more phylogenetic trees Mai et Mirarab 2018 (see details and install instructions)
- Requires Python 3.X (available on every Linux) and R 4.X to be installed
- Work in command line of your notebook (Linux, macOS), or on MetaCentrum

```
# On MetaCentrum load Python and R modules
module add python/3.9.12-gcc-10.2.1-rg2lpmk r/4.1.3-gcc-10.2.1-6xt26dl
git clone https://github.com/uym2/TreeShrink.git # Download TreeShrink
cd TreeShrink/
python3 setup.py install --user # Install it "classically"
# Go to directory with trees_good.nwk and run TreeShrink
~/.local/bin/run_treeshrink.py -h # See help
~/.local/bin/run_treeshrink.py -t trees_good.nwk -o treeshrink_exons_good \
  -O treeshrink_exons_good | tee treeshrink.log
ls -lha treeshrink_exons/ # Results
```

Outputs of TreeShrink

- Output (3 files) is saved into directory (in our case) `treeshrink_exons_good`
- File `treeshrink_exons_good.nwk` contains new list of phylogenetic trees in `NEWICK` which can be then used as an input for any species tree reconstruction software (ASTRAL, ...)
- `treeshrink_exons_good.txt` can be bit hard to read — it has one line for every tree in the input list and every line contains list of removed tips — if there is an empty line, no tip was removed from that particular tree; trees are not named, only in same order as in the original input file
- `treeshrink_exons_good_summary.txt` contains statistics for affected taxa

```
# Find out how many times particular sample was removed from the list of
# the trees
grep -o "\<[[:graph:]]\+\>" treeshrink_exons_good/treeshrink_exons_good.txt \
    | sort | uniq -c | sort -gr
```

Filtration tasks

Practice filtration of gene trees

1. Create PCoA at least from `trees_ml_exons.nwk`. Consider repeated running of PCoA for same input set of trees.
   - Try several different distance matrices (see slide 68). What are differences? Are all appropriate for all cases?
2. Run kdetrees on list of trees after removal of outliers detected by PCoA and create `trees_good.nwk` (or more similar files).
3. Process by TreeShrink at least `trees_ml_exons.nwk` and `trees_good.nwk`. Compare differences.
   - See also slide 65 for differences between both input files, i.e. prior running TreeShrink run:
     `sed -i 's/^[[:graph:]]\+ //' trees_ml_exons.nwk`
4. What are main differences between PCoA/kdetrees and TreeShrink?

Parsimony super tree

- Parsimony has plenty of implementation, example below is from `R` package `phangorn`
- Usage, principles and accuracy are more or less same...

```
# Compute parsimony super tree
?superTree # See help first...
tree.sp <- superTree(tree=trees.good, method="NNI", rooted=TRUE,
  trace=2, start=NULL, multicore=TRUE)
tree.sp # See details
tree.sp <- root(phy=tree.sp, outgroup=c("Riedelia-arfakensis_S49_L001",
  "Zingiber-officinale_S242_L001"), resolve.root=TRUE) # Root it
# Save parsimony super tree
write.tree(phy=tree.sp, file="parsimony_sp_tree.nwk")
# Plot parsimony super tree
plot.phylo(x=tree.sp, type="phylogram", edge.width=2,
  label.offset=0.01, cex=1.2)
add.scale.bar()
# Tune display of the tree...
```

Other options for species tree estimation in R

- Similar approach as `superTree` is implemented in `phytools::mrp.supertree`
- Distance-based tree reconstruction is in `ape::speciesTree`
- Coalescence model handling multiple individuals per species is in `phangorn::coalSpeciesTree`

```
?ape::speciesTree # See help...
?phytools::mrp.supertree
?phangorn::coalSpeciesTree
# All trees must be ultrametric - chronos scale them
trees.ultra <- lapply(X=trees.good, FUN=chronos, model="correlated")
class(trees.ultra) <- "multiPhylo"
# Calculate the species tree
tree.sp.mean <- speciesTree(x=trees.ultra, FUN=mean)
tree.sp2 <- mrp.supertree(tree=trees.good, method="optim.parsimony",
  rooted=TRUE)
```

ASTRAL and related tools

- ASTRAL-III and related tools like ASTER* or ASTRID-2 are popular easy to use methods to construct species tree out of set of gene trees
- From `https://github.com/smirarab/ASTRAL#installation` download `ZIP` archive, unpack `astral.5.7.8.jar` and edit `~/hybseq/bin/hybseq_6_sp_tree_2_run.sh` so that `java -jar /storage/…/astral.5.7.8.jar` points to correct location
- `~/hybseq/bin/hybseq_6_sp_tree_1_qsub.sh` must be edited before submission via `qsub` — change `WORKDIR` and `DATADIR`
- It goes to `DATADIR` and process all `*.nwk` files there
- The scripts can be easily edited to use with another similar tool
- Run the task (next slides), or download and inspect test data

Prepare lists of trees for ASTRAL and similar tools and submit job

```
# Create new directory and go there
cd ~/hybseq_course_zingibers/ && mkdir 5_sp_trees && cd 5_sp_trees/
# Copy to 5_sp_trees also results from R and TreeShrink, e.g.
cp ../4_gene_trees/trees_*.nwk ../4_gene_trees/treeshrink_*/*.nwk .
# Trees lists exported from R and in dir. 4_gene_trees contain gene name
# Assembly_10014 (Afr...eum_S118:0.0504,Afr...eta_S701:0.054,((((((Am...
# Remove gene (contig) names from lists of trees
sed -i 's/^[[:graph:]]\+ //' \
  trees_{cons,ml}_{exons,introns,supercontigs}.nwk
# Edit WORKDIR and DATADIR in ~/hybseq/bin/hybseq_6_sp_tree_1_qsub.sh and
# path to ASTRAL in ~/hybseq/bin/hybseq_6_sp_tree_2_run.sh and submit job
qsub -l walltime=4:0:0 -l select=1:ncpus=1:mem=4gb:scratch_local=1gb -m \
  abe ~/hybseq/bin/hybseq_6_sp_tree_1_qsub.sh
# Monitor running $USER's tasks with details
qstat -w -n -1 -u $USER # Last column contains machine name
# See your processes running the machine (from above list)
ssh exec_host "ps ux" # Replace exec_host by hostname!
```

Running species trees

Run species trees

1. Inspect scripts `~/hybseq/bin/hybseq_6_sp_tree_1_qsub.sh` and `~/hybseq/bin/hybseq_6_sp_tree_2_run.sh` and be sure to understand what they are doing, including syntax used.
2. Edit declarations of variables `WORKDIR` and `DATADIR` in `hybseq_6_sp_tree_1_qsub.sh` so that they point to correct locations.
3. Ensure to have correct path to ASTRAL `jar` file in `hybseq_6_sp_tree_2_run.sh`.
4. Run it to process all `*.nwk` files.
5. Inspect outputs and log files.

Other tasks with gene trees

For interested students

Tasks for advanced users with ASTRAL and relatives

1. Think how to run ASTRAL, ASTER*, ASTRID-2 or similar tool in your computer.
2. Edit `hybseq_6_sp_tree_2_run.sh` (and possibly `hybseq_6_sp_tree_1_qsub.sh`) to use different species tree builder.
3. Check parameters of ASTRAL, ASTER* and ASTRID-2 and other similar tools listed on their pages. What are differences? What are their advantages and disadvantages? Think about various possible settings.

Quick visualization of species trees I

- Individual species trees can be quickly visualized by simple **R** script processing in `for` loop each species tree input file
- Adjust code below and save it as `treeplot.r` in directory with species trees

```
library(ape)
library(scales)
fnames <- commandArgs(TRUE) # [1] file.nwk [2] file.png
tr <- read.tree(file=fnames[1])
tr <- root(phy=tr, outgroup="Zingiber-officinale_S242_L001",
  resolve.root=TRUE)
tr <- di2multi.phylo(phy=tr, tol=1e-08)
tr$edge.length[is.nan(tr$edge.length)] <- 1
png(filename=fnames[2], width=1500, height=1500, units="px", bg="white")
  plot.phylo(x=tr, type="phylogram", edge.color="blue", edge.width=4,
    cex=1.25, root.edge=TRUE, label.offset=0.1, align.tip.label=TRUE)
  title("Phylogenetic tree") # Ends on next slide ...
```

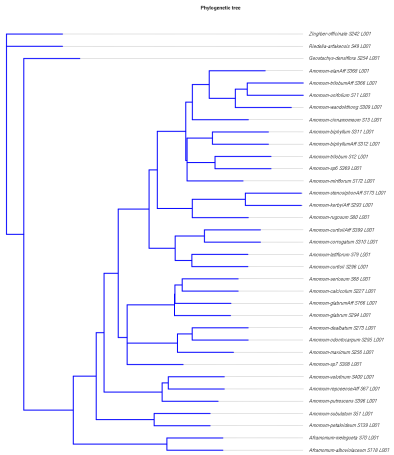
Quick visualization of species trees II

```
  edgelabels(text=round(x=tr$edge.length, digits=3), frame="rect",
    col="brown", bg=alpha(colour="white", alpha=0.5), cex=0.75)
  nodelabels(text=round(x=tr$node.label, digits=3), frame="rect",
    col="red", bg=alpha(colour="white", alpha=0.5), cex=1.2)
  axisPhylo()
  dev.off() # ... starts on previous slide
```

- Run following `for` loop to process `R` script from previous slide to get quick visualization of individual species trees
- Of course, this is very exemplary and especially graphical parameters must be highly adjusted according size of phylogeny etc.

```
module add r/4.1.3-gcc-10.2.1-6xt26dl # Load R. Packages: ape, scales
for L in sp_*.nwk; do echo "${L}"; R CMD BATCH --no-save --no-restore \
  "--args ${L} ${L%.nwk}.png" treeplot.r; done
# Note > " < around arguments to pass to R
```

Final species tree



Consensus network

- Available in `R` package `phangorn`
- Requires same set of tips in all trees

```
# See help
?consensusNet
# Compute consensus network
tree.net <- consensusNet(obj=trees.good, prob=0.25)
# Plot 2D or 3D
plot(x=tree.net, planar=FALSE, type="2D", use.edge.length=TRUE,
  show.tip.label=TRUE, show.edge.label=TRUE, show.node.label=TRUE,
  show.nodes=TRUE, edge.color="black", tip.color="blue") # 2D
plot(x=tree.net, planar=FALSE, type="3D", use.edge.length=TRUE,
  show.tip.label=TRUE, show.edge.label=TRUE, show.node.label=TRUE,
  show.nodes=TRUE, edge.color="black", tip.color="blue") # 3D
```

Phylonet

- Requires as input NEXUS file with PhyloNet commands (see below) — e.g. export from `R`:

```
ape::write.nexus(trees.good, file="trees_good.nex", translate=FALSE)
```

- Take `trees_good.nex` and append following code (or another command) to its end:

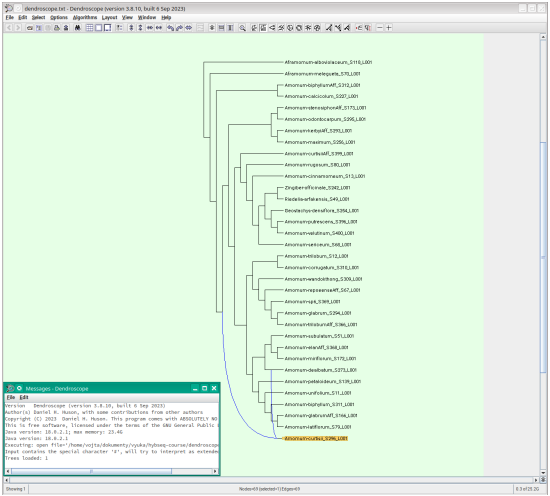
```
... # End of content of trees_good.nex
END; # Last fine of trees_good.nex - append the PHYLONET block below:
BEGIN PHYLONET;
InferNetwork_MP (all) 1 -b 50 -x 5 -pl 2 -di;
END;
```

```
# Download binary JAR file (ready to run)
wget https://phylogenomics.rice.edu/media/PhyloNet.jar
java -Xmx8g -jar PhyloNet.jar trees_good.nex | tee phylonet_exons.log
```

Running phylonet

- PhyloNet has various options how to create phylogenetic network
- Can be computationally very demanding, running long time — preferably run on MetaCentrum in standard batch job
- The `PHYLONET` section of the input NEXUS contains settings according to list of commands
- TreeID can be completely random, or simple consecutive sequence like GT0001–GT####
- PhyloNet can be computationally very demanding, calculating more than 1–3 reticulations can be unrealistic in terms of time needed...
- It **does not save output** file, the network in special NWK format for Dendroscope is on the end (after `Visualize in Dendroscope :`) — copy it from terminal or log file (e.g. `tee` from previous slide), extract and save as tiny TXT, which can be opened in Dendroscope

PhyloNet in Dendroscope



Installing phyparts and other tools

- Requires `maven` and several Python packages, installation can be complicated...

```
cd ~/bin/ || { mkdir ~/bin && cd ~/bin/; }
# Install Phyparts
git clone https://bitbucket.org/blackrim/phyparts.git
cd phyparts/
# Install dependencies
./mvn_cmdline.sh
# Install PhyParts_PieCharts
git clone https://github.com/mossmatters/MJPythonNotebooks.git
# Or on MetaCentrum (above steps are not needed then)...
module add phyparts/0.0.1
# Split list of trees into individual files
mkdir trees_good
split -a 4 -d -l 1 trees_good.nwk trees_good/trees_good_
ls trees_good/
# If applicable, open parsimony_sp_tree.nwk and remove 'Root' directive
```

Producing phyparts and phypartspiecharts.py outputs

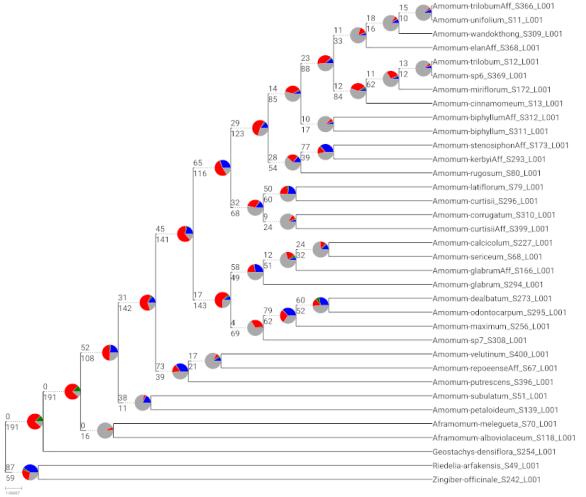
```
# Remove IQTREE ultrafast bootstrap values from gene trees
sed -i 's/\/[0-9]\{1,3\}//g' trees_good/trees_*
# Analysis with phyparts
java -jar \
  ~/bin/phyparts/target/phyparts-0.0.1-SNAPSHOT-jar-with-dependencies.jar \
  --help
java -jar \
  ~/bin/phyparts/target/phyparts-0.0.1-SNAPSHOT-jar-with-dependencies.jar \
  -a 1 -d trees_good -m parsimony_sp_tree.nwk -o trees_good_res -s 0.5 -v
# Copy phypartspiecharts.py to directory with trees
cp ~/bin/phyparts/MJPythonNotebooks/phypartspiecharts.py .
# See help for phypartspiecharts.py
python phypartspiecharts.py --help
# Pie chart: concordance (blue) top conflict (green), other conflict
# (red), no signal (gray). Run phypartspiecharts.py to get the graphics:
python phypartspiecharts.py --svg_name trees_good_res.svg \
  parsimony_sp_tree.nwk trees_good_res 219
```

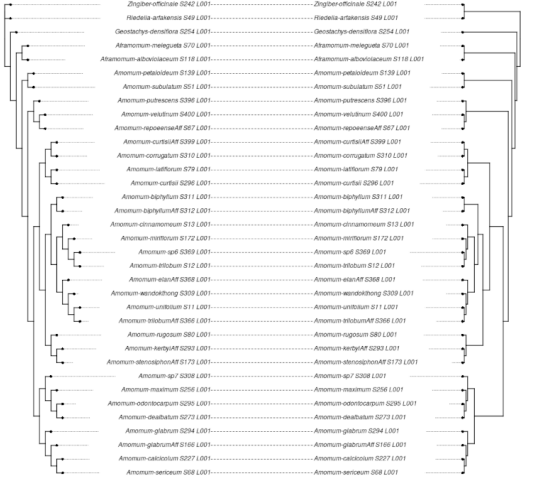
Comparing two trees — cophyloplots

- Slightly different implementation in `R` packages `ape` (`cophyloplot`) and `phytools` (`cophylo`)
- See help pages and play with graphical parameters

```
# We need 2 column matrix with tip labels
tips.labels <- matrix(data=c(sort(tree.sp[["tip.label"]]), sort
  (tree.sp2[["tip.label"]])), nrow=length(tree.sp[["tip.label"]]), ncol=2)
# Draw the tree, play with graphical parameters
# Click to nodes to rotate them to get better display
cophyloplot(x=tree.sp, y=tree.sp2, assoc=tips.labels, use.edge.length=
  FALSE, space=60, length.line=1, gap=2, type="phylogram", rotate=TRUE,
  col="red", lwd=1.5, lty=2)
# Slihtly better display in phytools::cophylo
trees.cophylo <- cophylo(tr1=tree.sp, tr2=tree.sp2, assoc=tips.labels,
  rotate=TRUE)
plot.cophylo(x=trees.cophylo, lwd=2, link.type="curved")
```

Phyparts and cophyloplot





Density tree

- The trees should be (otherwise plotting works, but may be more ugly) rooted, ultrametric and binary bifurcating
- Implementations are in `phangorn` (`densiTree`) and `phytools` (`densityTree`)

```
is.rooted.multiPhylo(trees.ultra) # rooted
is.ultrametric.multiPhylo(trees.ultra) # ultrametric
is.binary.multiPhylo(trees.ultra) # binary bifurcating
# See help page
?phangorn::densiTree
# Plotting density trees
densiTree(x=trees.ultra, scaleX=TRUE, col=rainbow(6), width=5, cex=1.5)
densiTree(x=trees.ultra, direction="upwards", scaleX=TRUE, width=5)
densiTree(x=trees.ultra, scaleX=TRUE, width=5, cex=1.5)
densiTree(x=trees.ultra[1:10], scaleX=TRUE, width=5, cex=1.25)
```

Different display for multiple trees

- `phytools::densiTree` requires same number of tips in all trees
- Note various ways how to select trees to display
- Nodes of the trees are not rotated (the display might be suboptimal)
- All trees must have same number of tips

```
# See help page
?phytools::densityTree
# Plotting density trees
densityTree(trees=c(tree.sp, tree.sp2), fix.depth=TRUE, lwd=4)
densityTree(trees=trees.ultra, fix.depth=TRUE, use.gradient=TRUE,
  alpha=0.5, lwd=4)
densityTree(trees=trees.ultra[1:3], fix.depth=TRUE, use.gradient=TRUE,
  alpha=0.5, lwd=4)
densityTree(trees=trees.ultra[c(2, 4, 6)], fix.depth=TRUE,
  use.gradient=TRUE, alpha=0.5, lwd=4)
```

This is just beginning of long journey...

- I presented common basic tools how to process HybSeq data and deal with various problems, but new tools keep emerging...
- Stay updated and keep trying new tools
- The scripts presented are not the only rigid way how to proceed, rather very general guideline, which should be subject of heavy modificcations according to your needs...

The end

Our course is over...

...I hope it was helpful for You...

...any feedback is welcomed...

...happy playing with the data...

...any final questions?

Typesetting using XeLaTeX on openSUSE GNU/Linux June 10, 2024