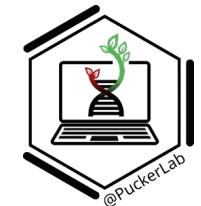
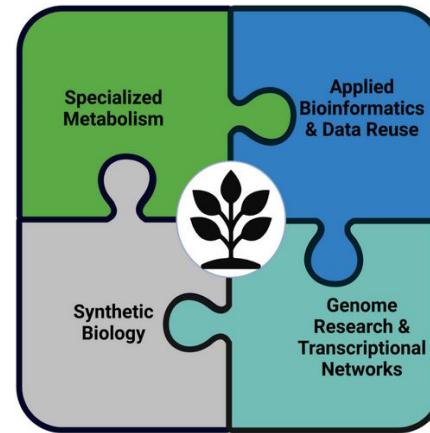


de.NBI course 2: From Gene Models to Biological Insights - Functional Annotation

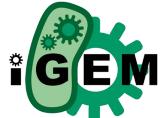
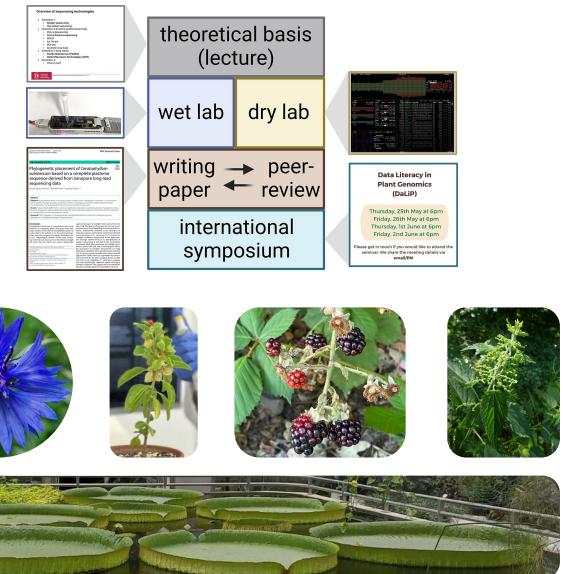




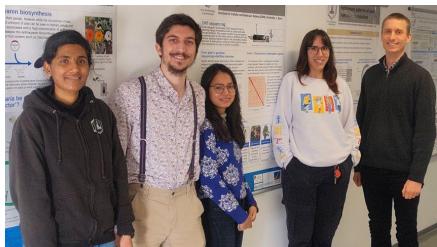
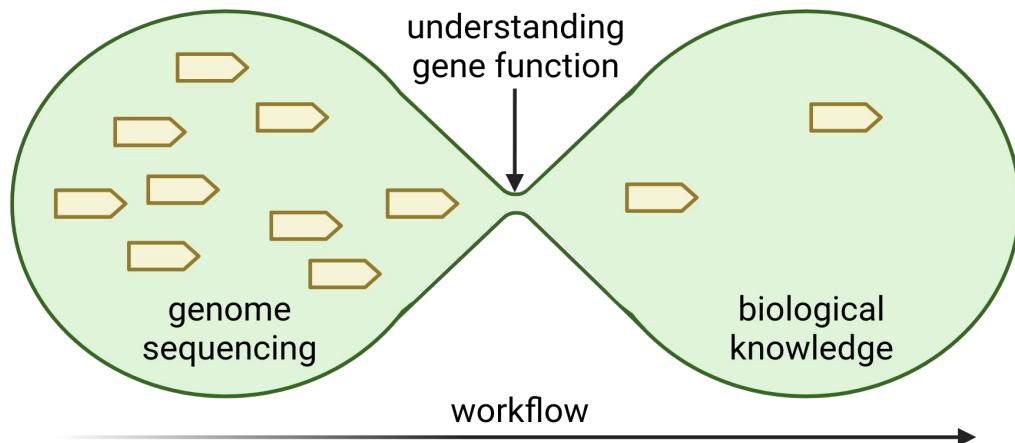
- Flavonoids: Anthocyanins, Flavonols, Flavones, Proanthocyanidins
- Withanolides
- Carotenoids
- ...



Tools: KIPES, NAVIP, MGSE, MYB/bHLH_annotator, DupyliCate...

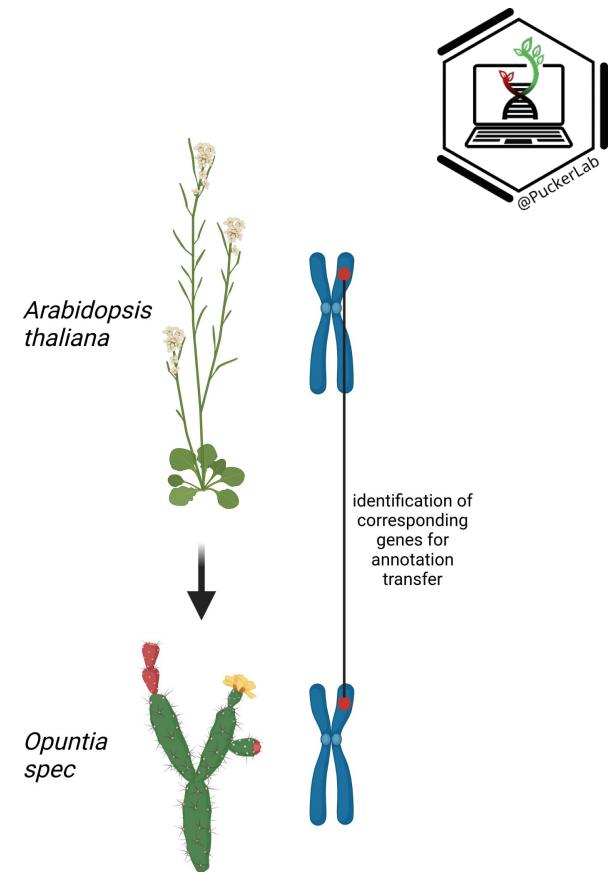


Course motivation

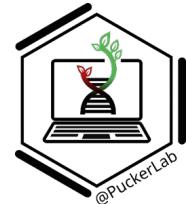


From L to R: Shakunthala, Samuel, Nancy, Julie, and Boas
Picture credits: Jakob Horz

From Gene Models to Biological Insights - Functional Annotation | Day 1 | 3



Availability of Slides

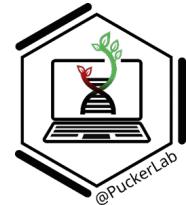


All materials will be shared with participants at the end of this course. Background information is available here:

<https://www.preprints.org/manuscript/202508.1176>

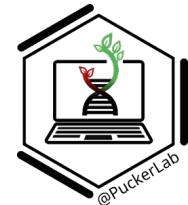
Slides → <https://github.com/bpucker/teaching/tree/master/deNBI2025>



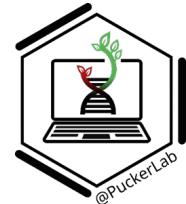


Today you will learn:

- Introduction to Virtual Machine
- Access to Virtual Machine
- Running BUSCO
- Transposable element annotation



- high performance computing => server farm



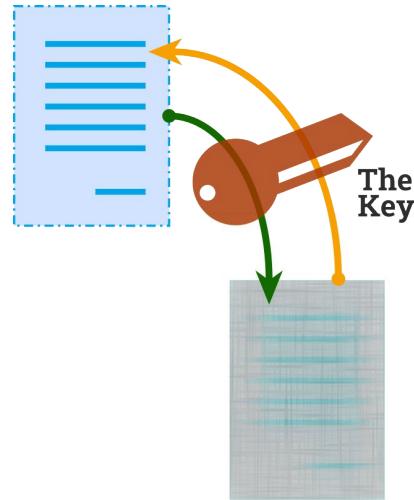
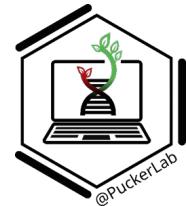
- high performance computing => server farm
- de.NBI offers VMs (VM = virtual machine)
 - running Linux
 - access through SSH
- de.NBI connected LifeScience AAI Account needed



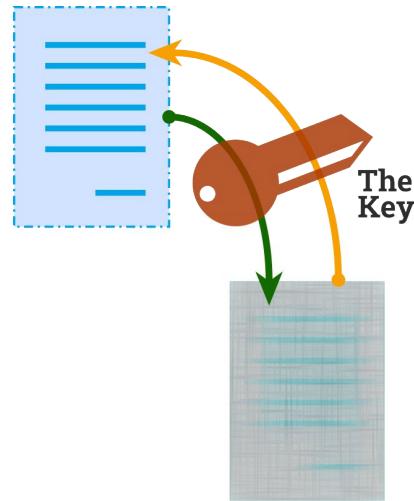
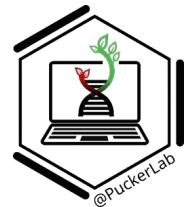
- high performance computing => server farm
- de.NBI offers VMs (VM = virtual machine)
 - running Linux
 - access through SSH
- de.NBI connected LifeScience AAI Account needed
- Tasks:
 1. Register: <https://cloud.denbi.de/wiki/registration/>
 2. Apply for the course de.NBI project:
<https://signup.aai.lifescience-ri.eu/fed/registrar/?vo=denbi&group=pbbcourse20251>

Who is new?

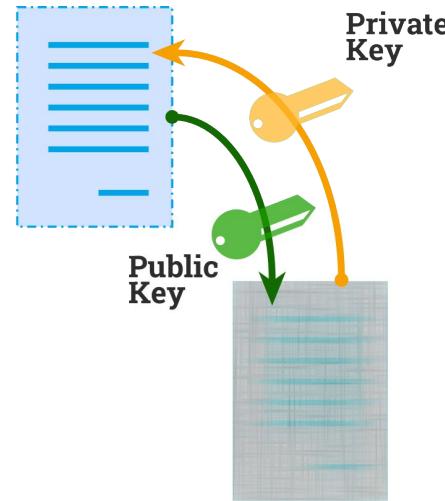




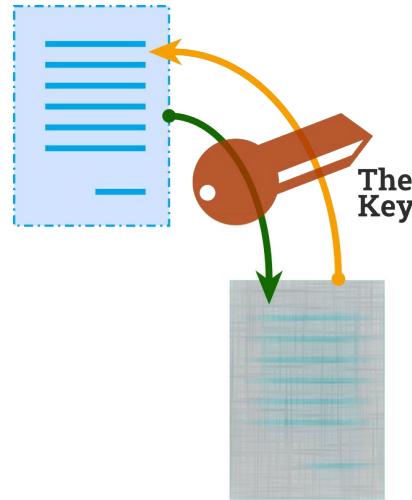
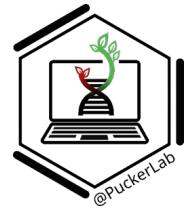
Symmetric encryption



Symmetric encryption



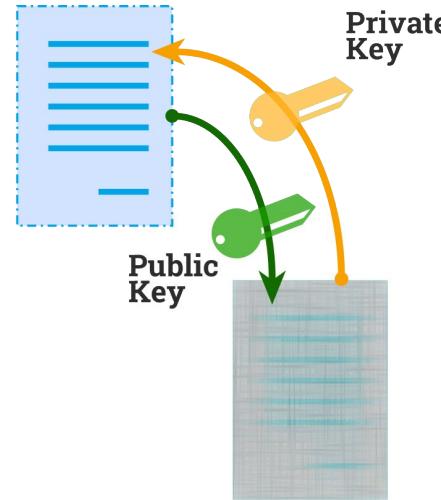
Asymmetric encryption



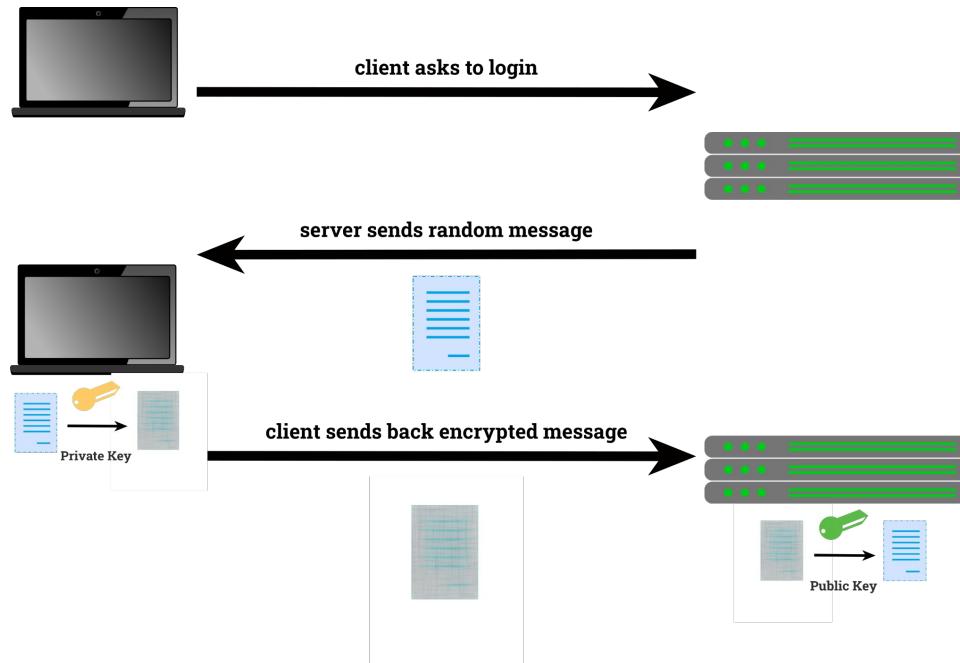
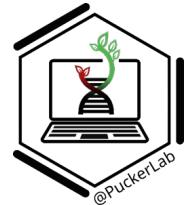
Symmetric encryption

ecdsa-sha2-nistp256

AAAAE2VjZHNhLXNoYTItbmlzdHAYNTYAAAAlbmlzdHAYNTYAAABBBJHT0aW+Q2bjkfVM7kN2ucaf8eltoalV6V0CDBgpWF+dKV1HKvGWWfjtYMOdLDUKIAzJc5Q6bKs6bkYiqqtUQ=



Asymmetric encryption





- high performance computing => server farm
- de.NBI offers VMs (VM = virtual machine)
 - running Linux
 - access through SSH
- de.NBI connected LifeScience AAI Account needed
- Tasks:
 1. Register: <https://cloud.denbi.de/wiki/registration/>
 2. Apply for the course de.NBI project:
<https://signup.aai.lifescience-ri.eu/fed/registrar/?vo=denbi&group=pbbcourse20251>
 3. create SSH key pair in de.NBI cloud portal and confirm unix user name



- SSH key might need the correct permissions
- needed information:
 - username
 - ip address/domain
 - port number (usually optional but mandatory for de.NBI VMs)
 - ssh private key
 - exemplary commands:

```
ssh -i /path/to/key username@ip-address -p port-number
```

```
ssh -i C:\Users\User\de.NBI_course\key.txt ubuntu@134.176.27.78 -p 23457
```

```
sam@dhcp068: ~$ ssh -i /home/TUBS_NC/13.DenBICloud_access_scripts/meckoni_ecdsa.txt ubuntu@134.176.27.78 -p 30235
[...]
de.NBI cloud
http://cloud.denbi.de
cloud@denbi.de

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-87-generic x86_64)

System information as of Sun Dec  7 14:28:58 UTC 2025

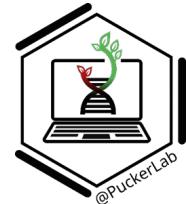
System load:  0.14      Processes:          422
Usage of /:   13.5% of 18.33GB   Users logged in:     0
Memory usage: 0%
Swap usage:   0%
IPv4 address for ens3: 192.168.0.235

Last login: Sun Dec  7 14:18:21 2025 from 192.168.0.143
ubuntu@snmtestdeletesoon-9b08e: ~$
```

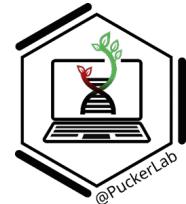


[https://github.com/bpucker/teaching
/tree/master/deNBI2025](https://github.com/bpucker/teaching/tree/master/deNBI2025)

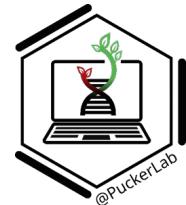




- Good structural annotation needed before functional annotation



- Good structural annotation needed before functional annotation
- BUSCO = Benchmarking Universal Single-Copy Orthologs



- Good structural annotation needed before functional annotation
- BUSCO = Benchmarking Universal Single-Copy Orthologs
- Universal genes -> expected to be in the genome
- Single-Copy -> expected to be present only once per haplotype



- Good structural annotation needed before functional annotation
- BUSCO = Benchmarking Universal Single-Copy Orthologs
- Universal genes -> expected to be in the genome
- Single-Copy -> expected to be present only once per haplophase
- if all BUSCO genes are present in the assembly => good assembly
- some might be natively missing due to gene loss in evolution

- exemplary commands:

list datasets:

```
sudo docker run --rm -v /vol/data:/vol/data ezlabgva/busco:v6.0.0_cv1 busco --list-datasets
```

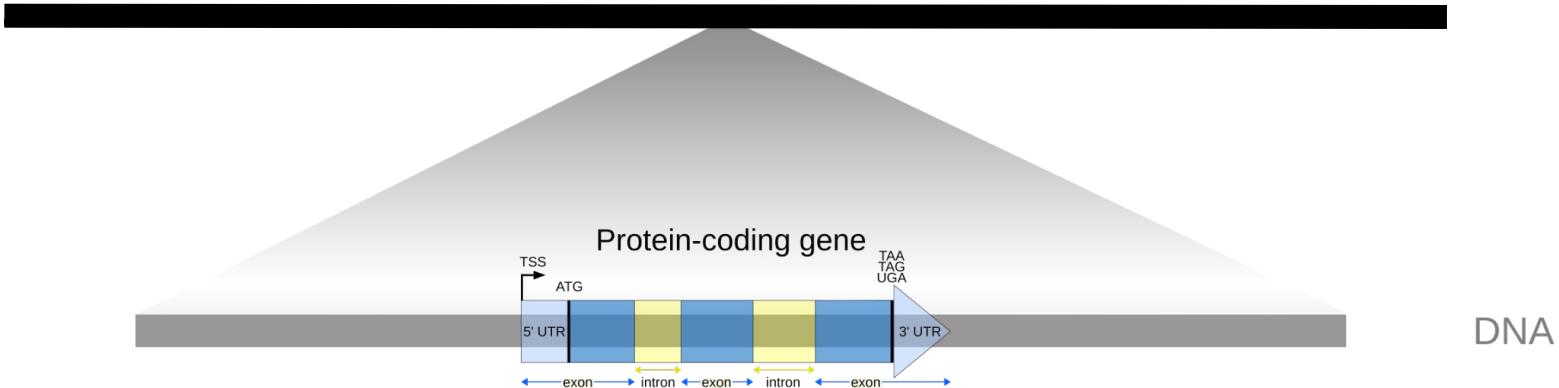
docker container name (will be automatically downloaded)

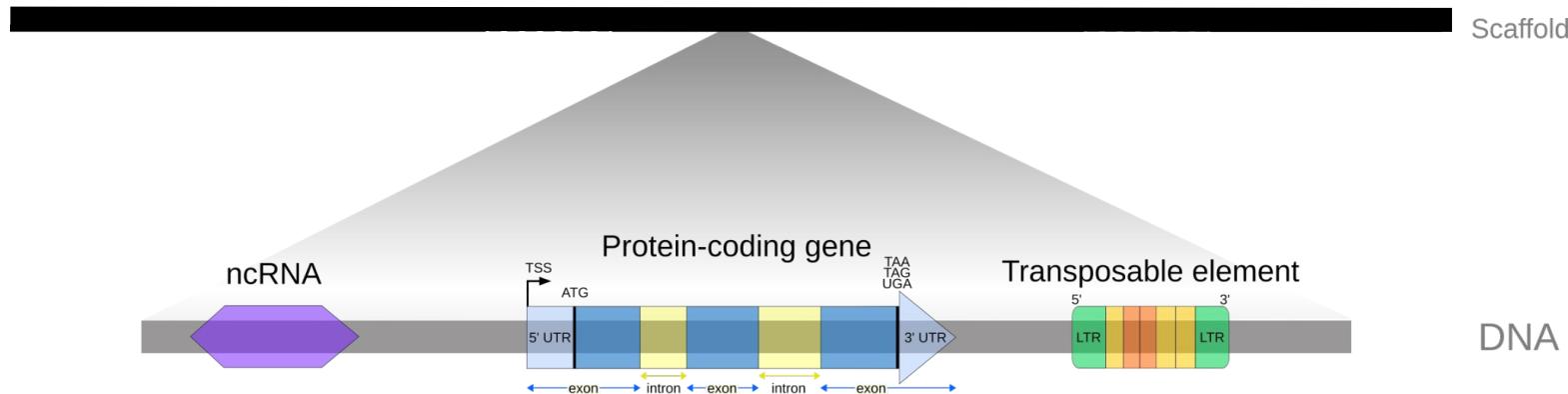
command that will be run in the
docker container starts here

important for the docker container to access files

start the docker container
input file
mode and lineage-dataset
output file

```
sudo docker run --rm -v /vol/data:/vol/data ezlabgva/busco:v6.0.0_cv1 \  
busco -i /vol/data/user/protein.fasta \  
-m proteins --cpu 13 -l brassicales_odb12 \  
--out_path /vol/data/user/BUSCO_pep/ -o busco_run_id --opt-out-run-stats
```



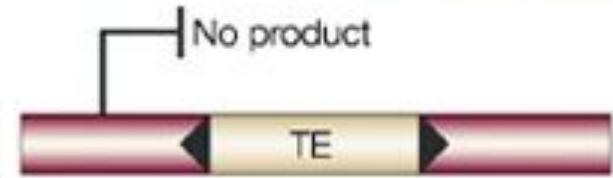


Not just protein-coding genes, but also non-coding DNA like
Transposable element (TEs) and non-coding RNAs (ncRNAs)

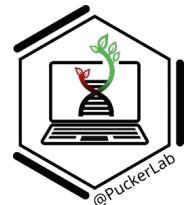
In plants, only ~1-10% protein-coding genes



- Discovered by Barbara McClintock in maize
- TEs responsible for large plant genome sizes
- Based on where they land, they dictate the regulation and impact

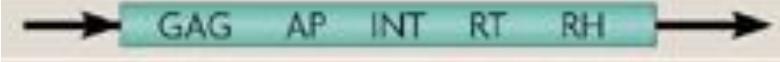


Classes of TEs

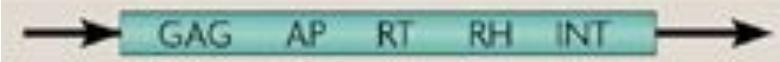


| Classification | Structure | TSD | Code | Occurrence |
|--|---------------|----------------------|----------|-------------|
| Order | Superfamily | | | |
| Class I (retrotransposons) | | | | |
| LTR | Copia | GAG AP INT RT RH | 4-6 | RLC P,M,F,O |
| | Gypsy | GAG AP RT RH INT | 4-6 | RLG P,M,F,O |
| | Bel-Pao | GAG AP RT RH INT | 4-6 | RLB M |
| | Retrovirus | GAG AP RT RH INT ENV | 4-6 | RLR M |
| | ERV | GAG AP RT RH INT ENV | 4-6 | RLE M |
| DIRS | DIRS | GAG AP RT RH YR | 0 | RYD P,M,F,O |
| | Ngoro | GAG AP RT RH YR | 0 | RYN M,F |
| | VIPER | GAG AP RT RH YR | 0 | RYV O |
| PLE | Penelope | RT EN | Variable | RPP P,M,F,O |
| LINE | R2 | RT EN | Variable | RIR M |
| | RTE | APE RT | Variable | RIT M |
| | Jockey | ORFI APE RT | Variable | RIJ M |
| | L1 | ORFI APE RT | Variable | RIL P,M,F,O |
| | I | ORFI APE RT RH | Variable | RII P,M,F |
| SINE | tRNA | | Variable | RST P,M,F |
| | 7SL | | Variable | RSL P,M,F |
| | 5S | | Variable | RSS M,O |
| Class II (DNA transposons) - Subclass 1 | | | | |
| TIR | Tc1-Mariner | Tase* | TA | DTT P,M,F,O |
| | hAT | Tase* | 8 | DTA P,M,F,O |
| | Mutator | Tase* | 9-11 | DTM P,M,F,O |
| | Merlin | Tase* | 8-9 | DTE M,O |
| | Transib | Tase* | 5 | DTR M,F |
| | P | Tase | 8 | DTP P,M |
| | PiggyBac | Tase | TTAA | DTB M,O |
| | PIF-Harbinger | Tase* ORF2 | 3 | DTH P,M,F,O |
| | CACTA | Tase ORF2 | 2-3 | DTC P,M,F |
| Crypton | Crypton | YR | 0 | DYC F |
| Class II (DNA transposons) - Subclass 2 | | | | |
| Helitron | Helitron | RPA Y2 HEL | 0 | DHH P,M,F |

LTR copia

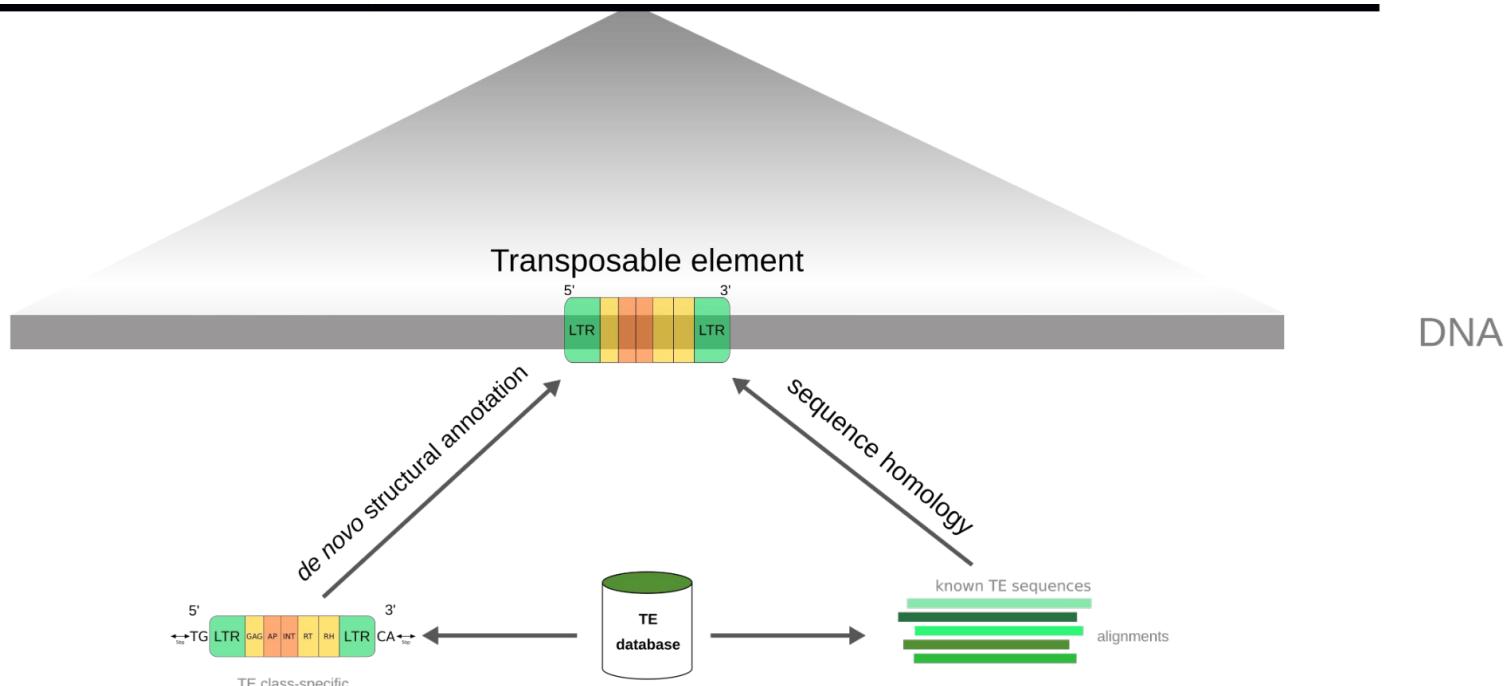
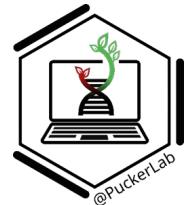


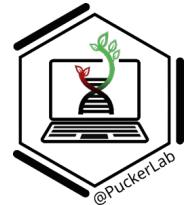
LTR gypsy



LINE L1







Specific tools

- LTR_FINDER/ LTR_HARVEST → LTRs
- TIR-LEARNER → TIRs
- Helitron scanner → Helitrons

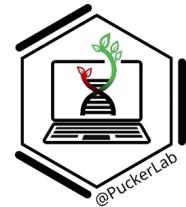
Specific tools

- LTR_FINDER/ LTR_HARVEST → LTRs
- TIR-LEARNER → TIRs
- Helitron scanner → Helitrons

EDTA (Extensive *de novo* TE annotator)

- pipeline combining above tools
- appropriate filtering
- Final TE library





Installing Miniconda

```
cd /vol/data/tools ← #Important not in home
```

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

```
bash Miniconda3-latest-Linux-x86_64.sh
```

```
# the above command will interactively ask 4 questions, first press ENTER to continue
```

```
# Do you accept license terms? [yes|no] → yes
```

```
#Miniconda3 will be installed in /home/ubuntu/miniconda3 or specify a different location → /vol/data/tools/Miniconda3
```

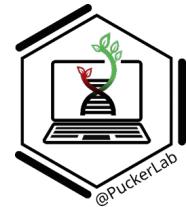
```
#Proceed with initialization [yes|no] → yes
```

```
source ~/.bashrc
```

```
conda config --add channels conda-forge
```

```
conda update -n base --all
```

```
conda install -n base mamba
```



Installing EDTA through conda/mamba

```
mamba create -n EDTA2.2 -c conda-forge -c bioconda -c r annosine2 biopython cd-hit coreutils genericrepeatfinder  
genometools-genometools glob2 tir-learner ltr_finder_parallel ltr_retriever mdust multiprocess muscle openjdk perl perl-text-soundex  
r-base r-dplyr regex repeatmodeler r-ggplot2 r-here r-tidyr tesorter samtools bedtools LTR_HARVEST_parallel HelitronScanner
```

Getting the latest version of EDTA from GitHub

```
cd /vol/data/tools; git clone https://github.com/oushujun/EDTA.git; cd EDTA; git checkout EDTA2; git branch
```

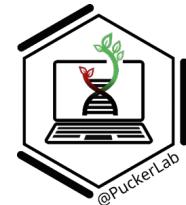
Running EDTA

```
cd /vol/data/user
```

```
mkdir EDTA_results; cd EDTA_results
```

```
conda activate EDTA2.2
```

```
perl /vol/data/tools/EDTA/EDTA.pl --genome /vol/data/DatasetB.genome.fasta --overwrite 1 --anno 1 --sensitive 1 --evaluate 1 --threads 13  
>>edta.log 2>&1 &
```

**DatasetB.genome.fasta**

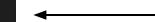
DatasetB.genome.fasta.mod

DatasetB.genome.fasta.mod.EDTA.TEanno.density_plots.pdf

DatasetB.genome.fasta.mod.EDTA.TEanno.gff3

DatasetB.genome.fasta.mod.EDTA.TEanno.sum

DatasetB.genome.fasta.mod.EDTA.TElib.fa

**DatasetB.genome.fasta.mod.EDTA.anno****DatasetB.genome.fasta.mod.EDTA.combine****DatasetB.genome.fasta.mod.EDTA.final**

DatasetB.genome.fasta.mod.EDTA.intact.fa

DatasetB.genome.fasta.mod.EDTA.intact.gff3

DatasetB.genome.fasta.mod.EDTA.raw

DatasetB.genome.fasta.mod.MAKER.masked

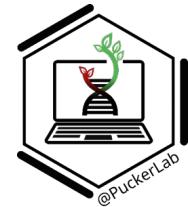
edta.log

DatasetB.genome.fasta.mod.RM2.raw.fa

edta2.log

DatasetB.genome.fasta.mod_divergence_plot.pdf

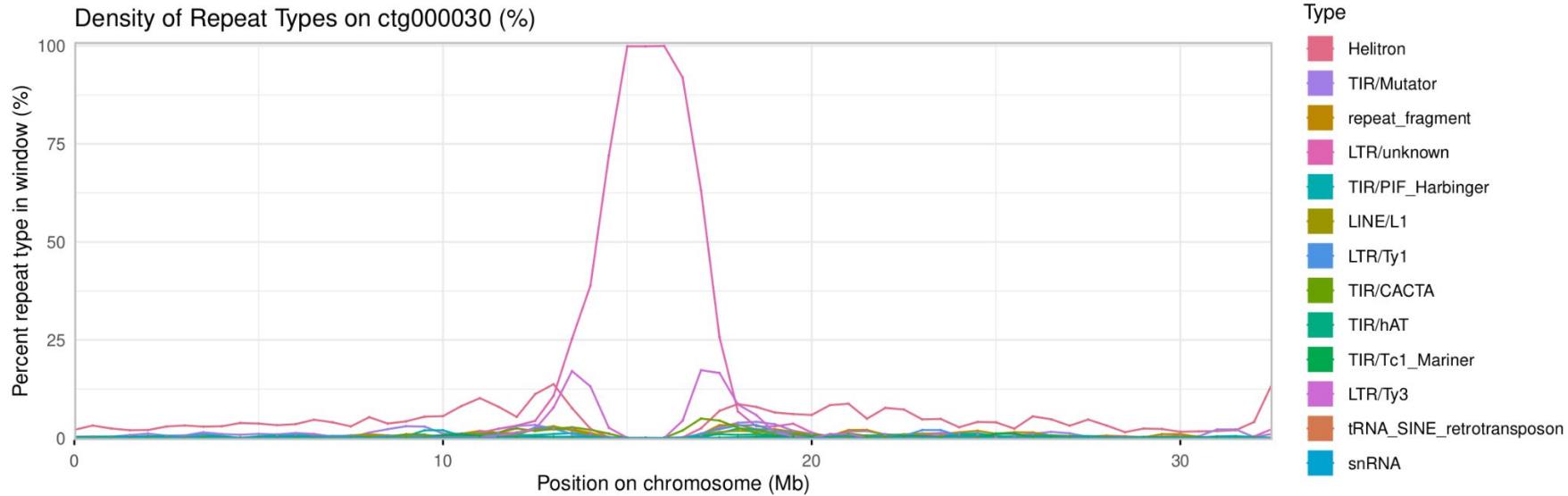
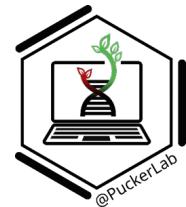
EDTA- output and results



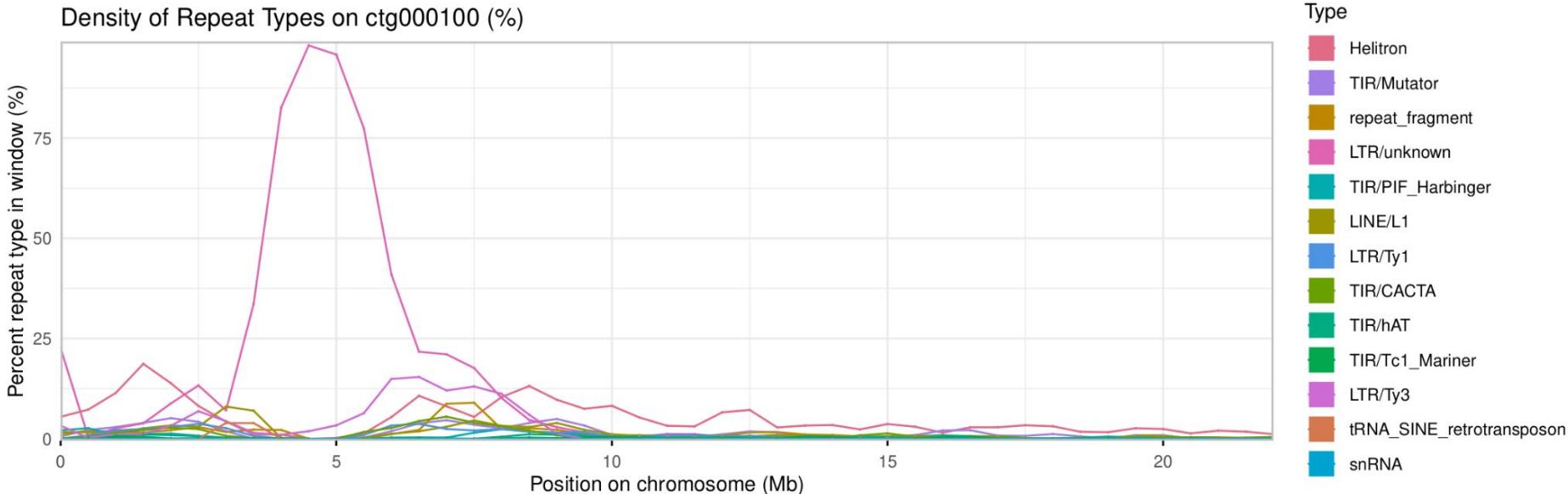
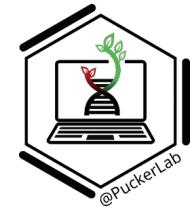
1. *.EDTA.TEanno.sum →
2. *.EDTA.TEanno.gff3
3. Density plots for top contigs

| Repeat Classes | | | |
|----------------------------|-------|----------|---------|
| ===== | ===== | ===== | ===== |
| Total Sequences: 13 | | | |
| Total Length: 134799993 bp | | | |
| Class | Count | bpMasked | %masked |
| ===== | ===== | ===== | ===== |
| LINE | -- | -- | -- |
| L1 | 1239 | 989157 | 0.73% |
| LTR | -- | -- | -- |
| Copia | 738 | 1002959 | 0.74% |
| Gypsy | 2081 | 3195026 | 2.37% |
| unknown | 5660 | 16713457 | 12.40% |
| SINE | -- | -- | -- |
| tRNA | 143 | 179011 | 0.13% |
| TIR | -- | -- | -- |
| CACTA | 1668 | 918659 | 0.68% |
| Mutator | 3170 | 1767375 | 1.31% |
| PIF_Harbinger | 1751 | 728452 | 0.54% |
| Tc1_Mariner | 539 | 259836 | 0.19% |
| hAT | 807 | 335497 | 0.25% |
| nonTIR | -- | -- | -- |
| helitron | 11058 | 5960229 | 4.42% |
| repeat_fragment | 3879 | 1580072 | 1.17% |
| ----- | | | |
| total interspersed | 32733 | 33629730 | 24.95% |
| ----- | | | |
| snRNA | 8 | 859 | 0.00% |
| ----- | | | |
| Total | 32741 | 33630589 | 24.95% |

Density plot



Density plot



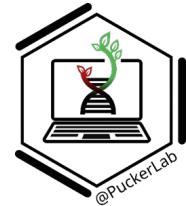


Today you learned:

- Accessing de.NBI VMs via SSH
- Working in a CLI environment
- Running BUSCO
- Identifying transposon elements

Tomorrow you will learn:

- tRNA annotation
- Running InterProScan
- KIPEs3, MYB_annotator
- Tree building

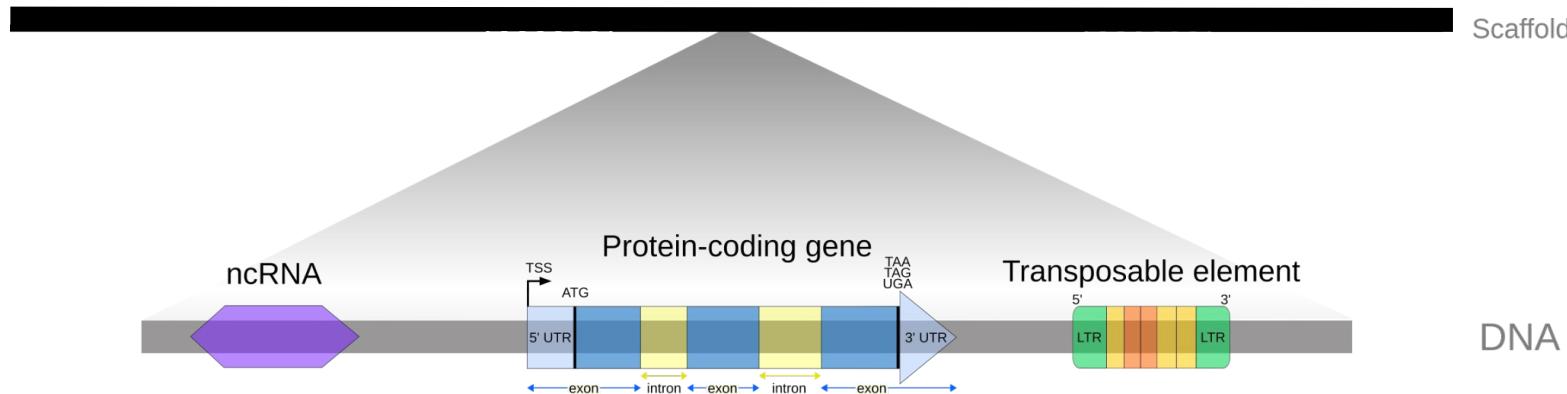


Yesterday you learned:

- Accessing de.NBI VMs via SSH
- Working in a CLI environment
- Running BUSCO
- Identifying transposon elements

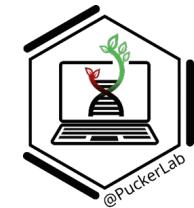
Today you will learn:

- tRNA annotation
- Running InterProScan
- KIPEs3, MYB_annotation
- Tree building



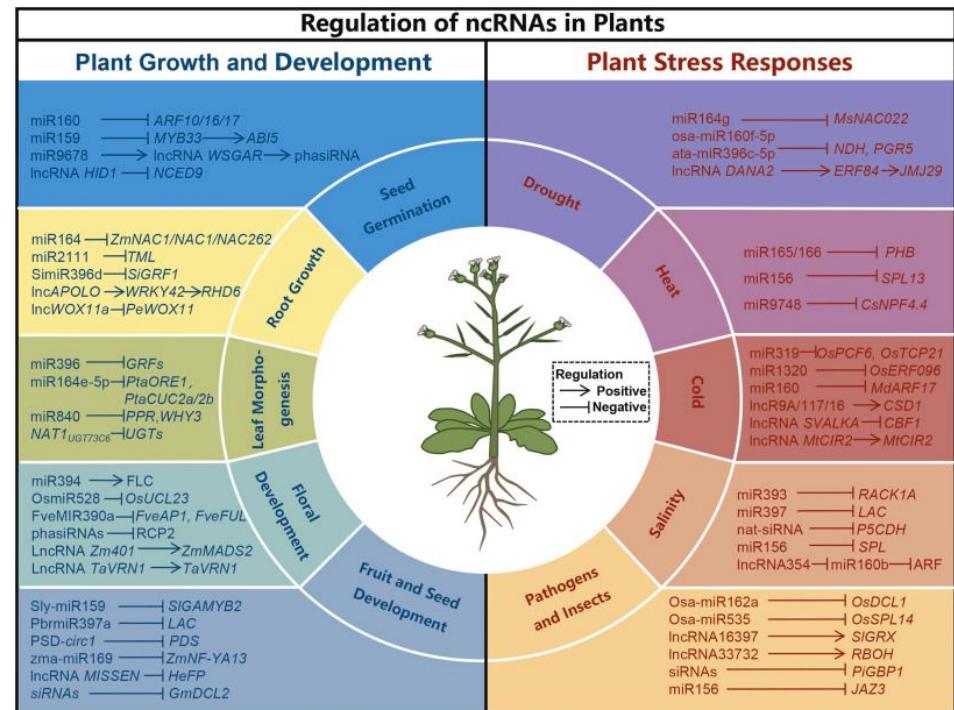
Not just protein-coding genes, but also non-coding DNA like
Transposable element (TEs) and non-coding RNAs (ncRNAs)

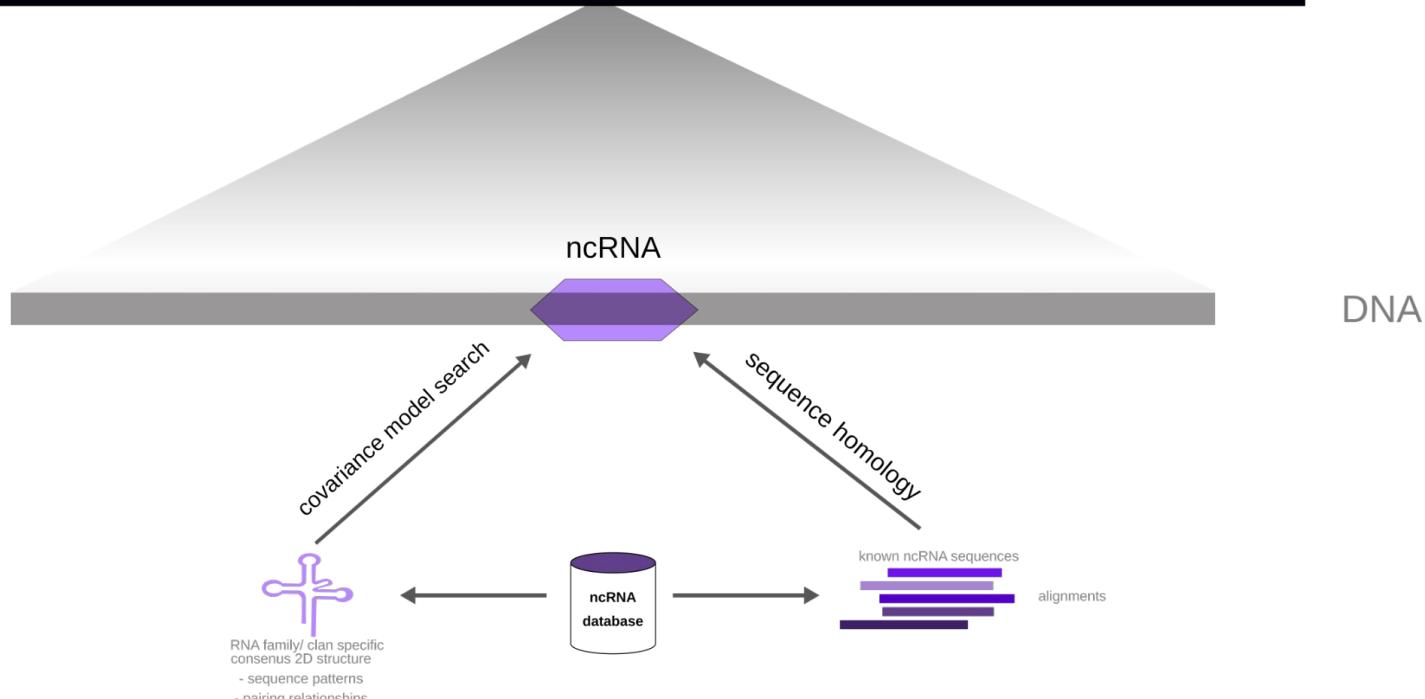
In plants, only ~1-10% protein-coding genes

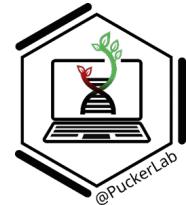


Types of Non-coding RNAs (ncRNAs)

- transfer RNAs (tRNAs)
- ribosomal RNAs (rRNA)
- small RNAs like microRNAs, siRNAs, snoRNAs, snRNAs







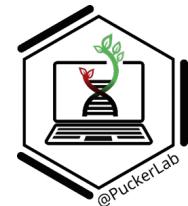
- **Infernal (INFERence of RNA ALignment)**

Identifies and classifies all ncRNAs in a genome sequence based on Rfam databases' calibrated covariance models

Specialized tools for different ncRNA classes

- tRNAs → tRNAscan-SE 2.0
- rRNA → RNAmmer, SSU-Align

Installing tRNAscan-SE 2.0



```
cd /vol/data/tools
```

location to install all tools

Get the latest release of tRNAscan-SE2.0 from GitHub
(<https://github.com/UCSC-LoweLab/tRNAscan-SE>)



UCSC-LoweLab / tRNAscan-SE (Public)

Releases 13

v2.0.12 Latest on Nov 16, 2022 + 12 releases

Releases / v2.0.12

v2.0.12 Latest

patrickaplchan released this Nov 16, 2022

Fixed anticodon detection issue for archaea

Assets 2

Source code (zip)

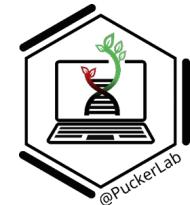
Source code (tar.gz)

right click
and copy link
address

```
wget https://github.com/UCSC-LoweLab/tRNAscan-SE/archive/refs/tags/v2.0.12.tar.gz #paste after wget in tools directory
```

```
tar -xvf v2.0.12.tar.gz #unarchive extract (-x), verbose (-v), unzip (-z), output in a new folder (-f)
```

```
cd tRNAscan-SE-2.0.12/
```



```
sudo ./configure
```

```
sudo make
```

```
sudo make install
```

Install dependency → Infernal

(<http://eddylab.org/infernal/>)

Installation where tRNAscan-SE 2.0 is installed

```
wget http://eddylab.org/infernal/infernal-1.1.5.tar.gz
```

```
tar -xvfz infernal-1.1.5.tar.gz
```

```
cd infernal-1.1.5
```

```
sudo ./configure
```

```
sudo make
```

```
sudo make install
```



Infernal: inference of RNA alignments

[infernal home](#) | [rfam database](#) | [eddy lab](#) |

Overview:

Infernal ("INFERence of RNA ALignment") is for searching DNA sequence databases for RNA structure and sequence similarities. It is an implementation of a special case secondary structure more than their primary sequence.

The latest release of Infernal is [1.1.5 \[7 Sep 2023\]](#).

Documentation:

- User's Guide [[PDF, 119 pages](#)].
- [README](#) from the current release.
- [Release notes](#) for the current release.

Reference:

- The recommended citation for using Infernal 1.1 is E. P. Nawrocki and S. R. Eddy, [Infernal 1.1: 100-fold faster RNA homology searches](#), *Bioinformatics* 29:2933–2938.

Downloads:

- The current source code: [[Infernal 1.1.5 source tarball, 31.3 MB](#)]

right click
and copy link
address



```
(base) ubuntu@lilyamir-4e1bc:~$ tRNAscan-SE

tRNAscan-SE 2.0.12 (Nov 2022)

FATAL: No sequence file(s) specified.

Usage: tRNAscan-SE [-options] <FASTA file(s)>

Scan a sequence file for tRNAs
  -- default: use Infernal & tRNA covariance models
  with eukaryotic sequences
  (use -B, -A, -M, -O or -G to scan other types of sequences)
```

Example command:

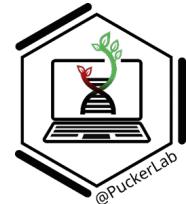
```
tRNAscan-SE -o Athaliana_tRNA.out -m Athaliana_tRNA.stats /vol/data/DatasetB.genome.fasta
```



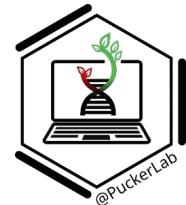
1. Athaliana_tRNA.stats

| | |
|---|-----|
| tRNAs decoding Standard 20 AA: | 881 |
| Selenocysteine tRNAs (TCA): | 0 |
| Possible suppressor tRNAs (CTA,TTA,TCA): | 0 |
| tRNAs with undetermined/unknown isotypes: | 9 |
| Predicted pseudogenes: | 18 |
| <hr/> | |
| Total tRNAs: | 908 |
| tRNAs with introns: | 70 |

2. Athaliana_tRNA.out → comprehensive result file with positions of predicted tRNAs in the genome sequence

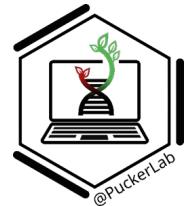


- What is the function of a gene
- Knockout experiments for all genes are expensive and time consuming
- Annotation transfer - orthologs are assumed to have the same function



- InterPro - database integrating predictive info about protein function from a number of partner resources - CATH, CDD, PANTHER, Pfam and so on
- Hosted and maintained by EMBL-EBI
- InterProScan5 - software to perform functional annotation of proteins
- Integrated with the InterPro database





Installing pre-requisites

- java -version

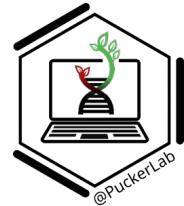
```
openjdk version "11.0.4" 2019-07-16
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.4+11)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.4+11, mixed mode)
```

If not, do:

sudo apt update

sudo apt install openjdk-11-jdk **and check** java -version

sudo update-alternatives --config java **(select java-11)**



#Set JAVA_HOME for OpenJDK 11

- readlink -f /usr/bin/java | sed "s:/bin/java::"
- /usr/lib/jvm/java-11-openjdk-amd64
- echo 'export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64' >> ~/.bashrc
- echo 'export PATH=\$JAVA_HOME/bin:\$PATH' >> ~/.bashrc
- source ~/.bashrc



Step1: fetch InterProScan tarball

```
cd /vol/data/tools
```

```
wget
```

```
https://ftp.ebi.ac.uk/pub/software/unix/iprscan/5/5.76-107.0/interproscan-5.76-107.0-64-bit.tar.gz
```

Step 2: fetch checksum

```
wget
```

```
https://ftp.ebi.ac.uk/pub/software/unix/iprscan/5/5.76-107.0/interproscan-5.76-107.0-64-bit.tar.gz.md5
```

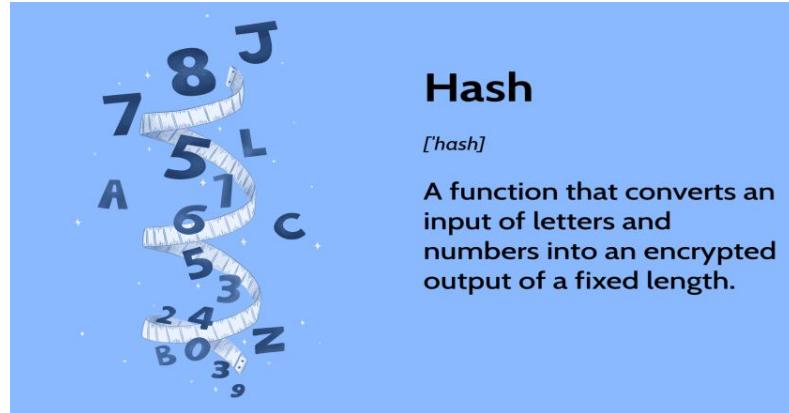
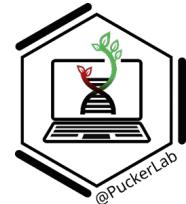
Step 3:#checksum to confirm the download was successful:

```
md5sum -c interproscan-5.76-107.0-64-bit.tar.gz.md5
```

```
# Must return *interproscan-5.76-107.0-64-bit.tar.gz: OK*
```

Step 4:#Extract the tar ball

```
tar -pxvzf interproscan-5.76-107.0-*bit.tar.gz
```



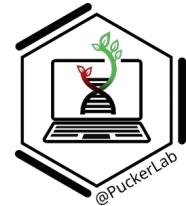
Hash

[*hash*]

A function that converts an input of letters and numbers into an encrypted output of a fixed length.

- Message Digest Algorithm 5
- Verifies file authenticity and integrity
- Simply, to ensure the data fetched is correct and complete

InterProScan5 setting up



#Run the setup script before the 1st time you run InterProScan

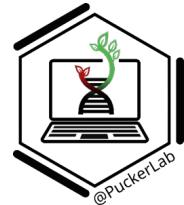
```
cd interproscan-5.76-107.0
```

```
python3 setup.py -f interproscan.properties
```

#Check

```
./interproscan.sh
```

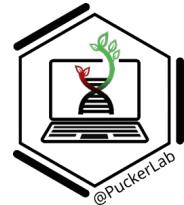
Preprocessing PEP file



- InterProScan complains about *
- Remove * before running InterProScan
- #Command

```
cd /vol/data/InterProScan5_input
```

```
sed 's/\/*//g' Arabidopsis_thaliana.pep.fasta > Arabidopsis_thaliana_cleaned.pep.fasta
```



#test run

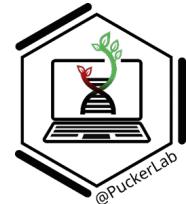
```
cd /vol/data/tools/interproscan-5.76-107.0  
./interproscan.sh -i test_all_appl.fasta -f tsv -dp  
./interproscan.sh -i test_all_appl.fasta -f tsv
```

#model command

```
./interproscan.sh -i /vol/data/InterProScan5_input/Arabidopsis_thaliana_cleaned.pep.fasta -appl  
Pfam,PANTHER -b /vol/data/user/Athaliana_interpro -f tsv --goterms --pathways --cpu 12 >>  
/vol/data/user/interpro.log 2>&1 &
```

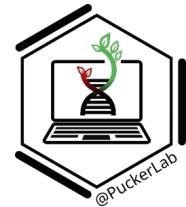


| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|-------------------|----------------------------------|-----------------|---------------------|-----------|-------------------------------------|------|-------|----------|------|-----------------|-----------|---|-----------------------|---|
| Analysis | | | | | Signature description | | Start | e-value | Date | IPR description | | Pathways annotation | | |
| Athal23200.2 | 9e6864249995acccebc284a5168cc585 | 232 | PANTHER | PTHR43874 | TWO-COMPONENT RESPONSE REGULATOR | 7 | 166 | 4.10E-52 | T | ##### | IPR045279 | Two-component response regulator ARR-like | GO:0009736 (InterPro) | - |
| protein accession | sequence md5 digest | sequence length | Signature accession | | | Stop | True | | | IPR accession | | | GO annotation | |



- Python script developed in-house
- Constructs functional annotation file for a given set of peptide sequences
- Uses functional annotation of *Arabidopsis thaliana* and BLAST to transfer the functions across





#Step 1: Get the BLAST tar ball

```
cd /vol/data/tools
```

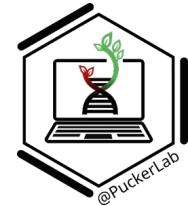
```
wget https://ftp.ncbi.nlm.nih.gov/blast/executables/LATEST/ncbi-blast-2.17.0+-x64-linux.tar.gz
```

#Step 2: Decompress the tar ball

```
tar -xzvf ncbi-blast-2.17.0+-x64-linux.tar.gz
```

```
cd ncbi-blast-2.17.0+/bin;ls
```

BLAST installation



#Step 3: add mafft to the bash path (OPTIONAL)

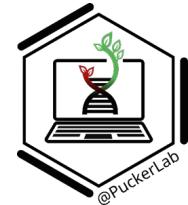
```
nano ~/.bashrc
```

#Step 4: Add this line at the end of the bashrc file:

```
export PATH=$PATH:/vol/data/tools/ncbi-blast-2.17.0+/bin
```

#Step 5: update/ reload the bashrc file

```
source ~/.bashrc
```



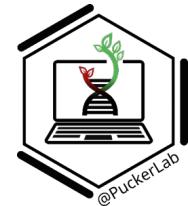
#Step 1: Fetch the script from GitHub

```
wget
```

```
https://raw.githubusercontent.com/bpucker/PBBtools/refs/heads/main/collection/construct\_anno.py
```

#Step 2: Read the script usage

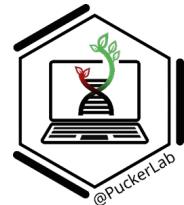
```
python3 construct_anno.py
```



#Step 3: Run the script

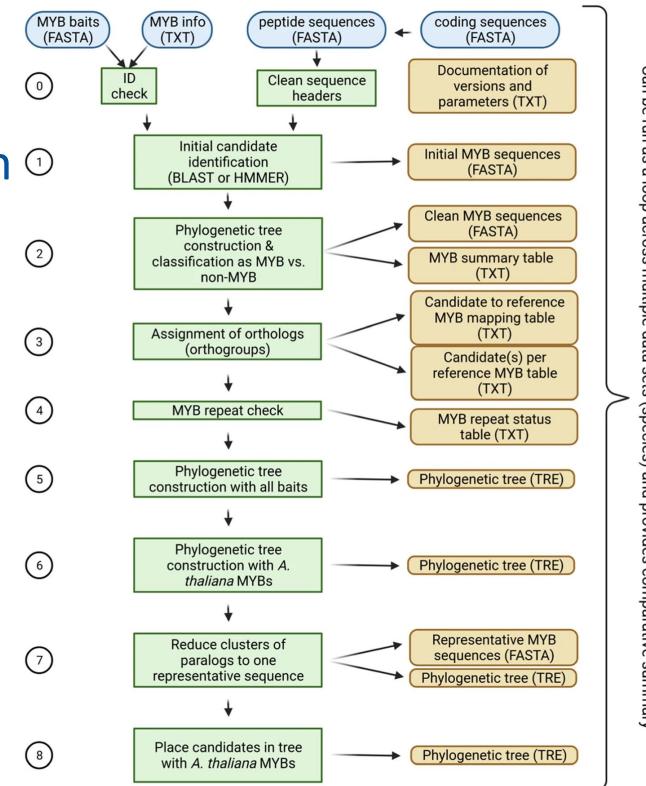
```
python3 construct_anno.py --in /vol/data/construct_anno_input/Carica_papaya.pep.fasta --ref  
/vol/data/InterProScan5_input/Arabidopsis_thaliana.pep.fasta --anno  
/vol/data/construct_anno_input/Athaliana_functional_annotation.tsv --out  
/vol/data/transfer_annotation --cpu 12
```

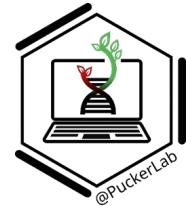
Interested in specific genes or pathway?





- Python based pipeline for automatic identification and functional annotation of MYBs
- Basis
 1. Domain and motif identification
 2. Phylogenetic analysis with well-characterized sequences





Installation

```
git clone https://github.com/bpucker/MYB_annotator.git
```

Requirements:

dendropy → pip3 install dendropy

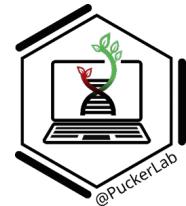
mafft → sudo apt install mafft

fasttree → sudo apt install fasttree

hmmer → sudo apt install hmmer

Print usage:

```
python3 /vol/data/tools/MYB_annotator/MYB_annotator.py
```



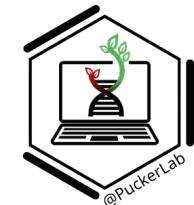
Example command:

```
python3 /vol/data/tools/MYB_annotator/MYB_annotator.py --baits\  
/vol/data/tools/MYB_annotator/MYB_baits.fasta --info \  
/vol/data/tools/MYB_annotator/MYB_baits.txt --out /vol/data/MYB_annotator/ \  
--subject /vol/data/GeMoMa/filter1/final/proteins.fasta --ath \  
/vol/data/tools/MYB_annotator/AthMYBs.fasta --refmybs \  
/vol/data/tools/MYB_annotator/AthRefMYBs.txt
```

RESULTS directory

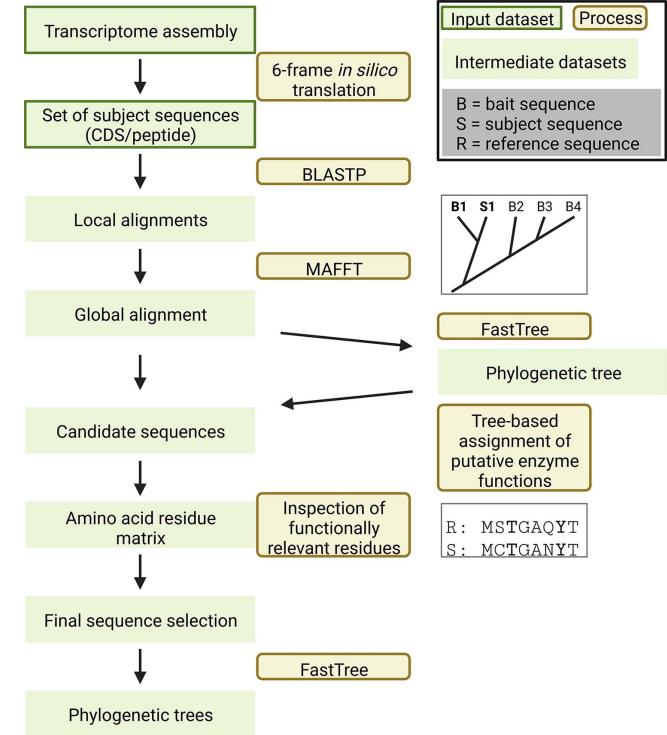
- 03b_new_2_ref_myb_mapping_file.txt
- *_tree.tre

| NewMYB | RefMYB | EdgeDistance | PatristicDistance | Annotation |
|--------------|-----------------------|--------------|-------------------|--|
| Athal00230.1 | At4g38620-At2R-MYB004 | 2 | 1e-08 | repressor phenylpropanoid, sinapate, lignin |
| Athal00327.1 | At4g37780-At2R-MYB087 | 2 | 1e-08 | axillary meristem, root growth |
| Athal00390.1 | At4g37260-At2R-MYB073 | 2 | 0.0 | stress response, hormone signaling |
| Athal00658.1 | At4g34990-At2R-MYB032 | 2 | 1e-08 | repressor phenylpropanoid, sinapate, lignin |
| Athal00850.1 | At4g33450-At2R-MYB069 | 2 | 0.0 | cell wall, lignin, seed oil, axillary meristem |
| Athal00850.2 | At4g33450-At2R-MYB069 | 2 | 0.0 | cell wall, lignin, seed oil, axillary meristem |
| Athal00928.1 | AT4G32730.1-At3R-MYB1 | 2 | 0.0 | cell cycle control |
| Athal00928.2 | AT4G32730.1-At3R-MYB1 | 2 | 0.0 | cell cycle control |
| Athal01516.1 | At4g28110-At2R-MYB041 | 2 | 1e-08 | defense, stress response |
| Athal01665.1 | At4g26930-At2R-MYB097 | 2 | 1e-08 | anther development, stress response |
| Athal01827.1 | At4g25560-At2R-MYB018 | 2 | 1e-08 | photomorphogenesis |
| Athal02170.1 | At4g22680-At2R-MYB085 | 2 | 1e-08 | SCW, lignin |



Knowledge-based Identification of Pathway Enzymes (KIPeS3)

- Automatic identification and annotation of any biological pathway of interest
- Flavonoid and Carotenoid biosynthesis pathway well established
- Basis: (a) orthology, and (b) conserved domains and residues
- Available as a command line-tool and on webserver:
<https://pbb-tools.de/>



Installing KIPEs (<https://github.com/bpucker/KIPEs>)

```
cd /vol/data/tools
```

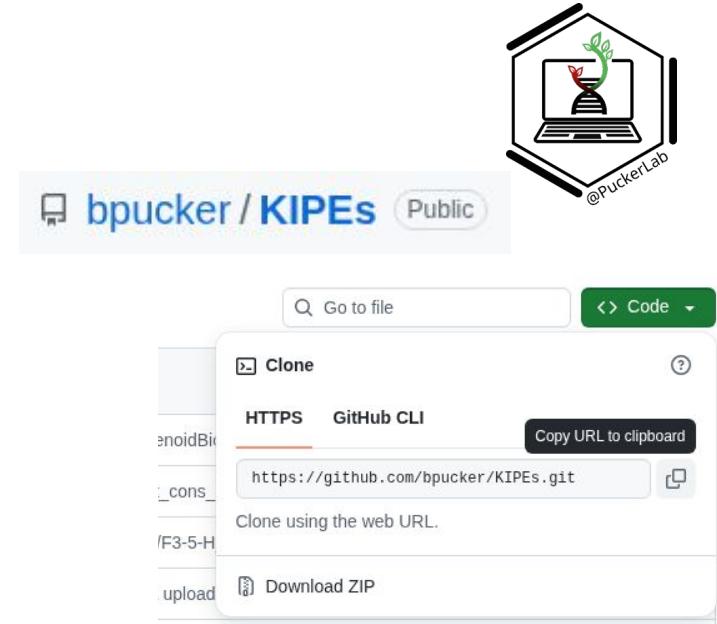
```
git clone https://github.com/bpucker/KIPEs.git
```

```
cd KIPEs
```

KIPEs3.py

carotenoid

flavonoid

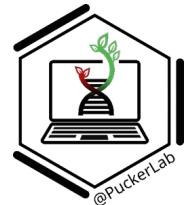


cd baits/

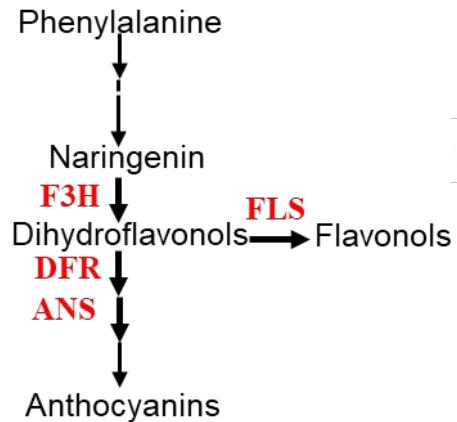
```
(base) ubuntu@pga2024-02ba6:/vol/data/tools/KIPEs/baits$ ls  
3AT.fasta    4CL.fasta    A3RT.fasta    ABCC.fasta    ANR.fasta    BZ2.fasta    CHI1.fasta    CHS.fasta    F2H.fasta      F3-5  
-H-b.fasta   F3GT.fasta   FLS.fasta    FNS2.fasta   MATE.fasta   PAL.fasta    TT15.fasta   TT9.fasta  
3MAT.fasta   5MAT.fasta   A5GT.fasta   AHA10.fasta  ANS.fasta   C4H.fasta    CHI2.fasta   DFR.fasta   F3-5-H-a.fasta  F3-H  
.fasta       F3H.fasta    FNS1.fasta   LAR.fasta   OMT.fasta_  TT10.fasta   TT19.fasta   UGT72L1.fasta
```

cd residues/

```
(base) ubuntu@pga2024-02ba6:/vol/data/tools/KIPEs/residues$ ls  
3AT.res      A3RT.res     ANR.res     CHI1.res     F2H.res      F3GT.res     FNS2.res     PAL.res     TT9.res  
3MAT.res     A5GT.res     ANS.res     CHI2.res     F3-5-H-a.res F3H.res     LAR.res     TT10.res    UGT72L1.res  
4CL.res      ABCC.res     BZ2.res     CHS.res     F3-5-H-b.res FLS.res     MATE.res    TT15.res  
5MAT.res     AHA10.res    C4H.res     DFR.res     F3-H.res     FNS1.res     OMT.res    TT19.res
```

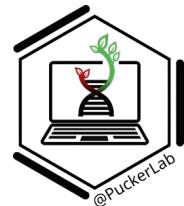


Pathway of interest

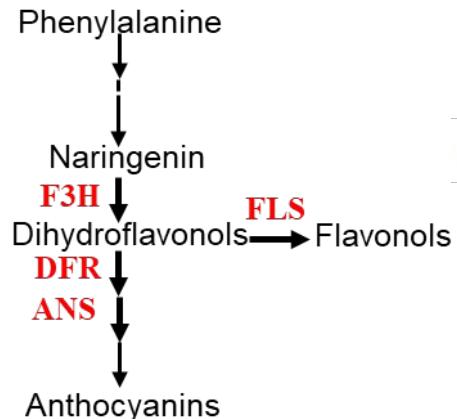


Collection of baits

| | F3H | FLS | DFR |
|-------|-----------------|------|------|
| Bait1 | MWNIILCIVGLV... | | |
| Bait2 | | | |
| Bait3 | | | |



Pathway of interest

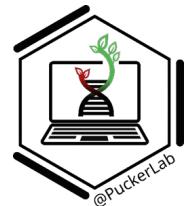


Collection of baits

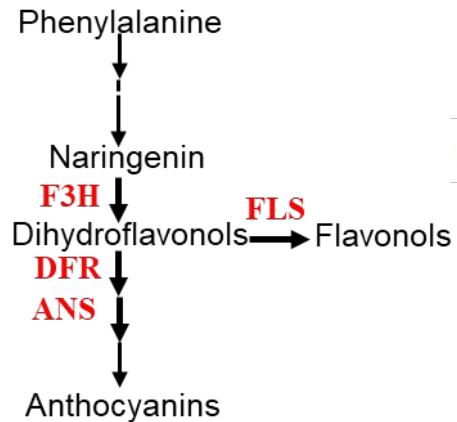
| | F3H | FLS | DFR |
|-------|-----------------|------|------|
| Bait1 | MWNIILCIVGLV... | | |
| Bait2 | | | |
| Bait3 | | | |

Ortholog sequences

| | | | |
|-----------|---|----|---|
| Bait 1 | 1 MWNIILCIVGLVVVGITWWVYRWRNPKCRSVLPPGSMGLPLIG | 43 | |
| Bait 2 | | | |
| Subject 1 | 3 MWDIILCILGVVVVGITHWVYRWRNPCKGVLPPGSMGLPLIG | 45 | ✓ |
| Subject 2 | 2 MWNIILCIVGLVVVGITWWVYKLRNPKCRSVLPPGSMGLPLIG | 44 | ✓ |
| Subject 3 | 1 MWNIILCIVGLVVVGITHWVYRLSPGECKGVLPPIGSMLPLIG | 43 | ✓ |
| Subject 4 | 1 MSSSCPSKVSRLFLSEDSKETISTKCKELIATLPKLDPPHPTY | 43 | ✗ |



Pathway of interest



Collection of baits

| | F3H | FLS | DFR |
|-------|-----------------|------|------|
| Bait1 | MWNIILCIVGLV... | | |
| Bait2 | | | |
| Bait3 | | | |

Predicted sequences

| | F3H | FLS | DFR |
|-------|-----------------|------|------|
| Subj1 | MWNIILCIVGLV... | | |
| Subj2 | MWDIILCILGVV... | | |
| Subj3 | MWNIILCIVGLV... | | |

Ortholog sequences

Bait 1 1 MWNIILCIVGLVVVGITWWVYRWRNPKCRSVLPPGSMGLPLIG 43

Bait 2

Subject 1 3 MWDIILCILGVVVVGITHWVYRWRNPCKGVLPPGSMGLPLIG 45 ✓

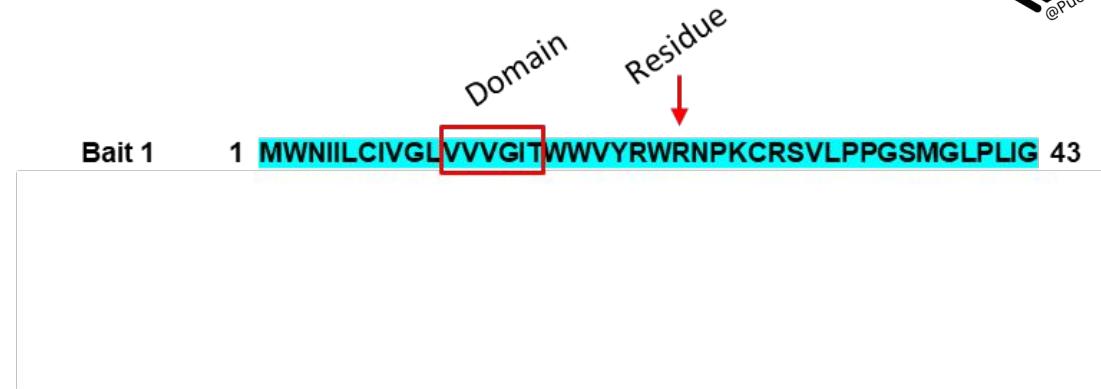
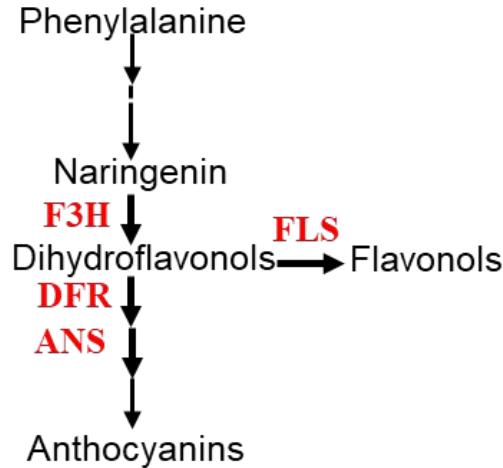
Subject 2 2 MWNIILCIVGLVVVGITWWVYKLRNPKCRSVLPPGSMGLPLIG 44 ✓

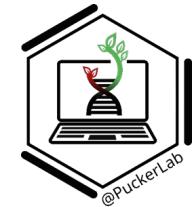
Subject 3 1 MWNIILCIVGLVVVGITHWVYRLSPGECKGVLPAGSMGLPLIG 43 ✓

Subject 4 1 MSSCPSKVSRLFLSEDSKETISTKCKELIATLPKLDPPHPTY 43 ✗

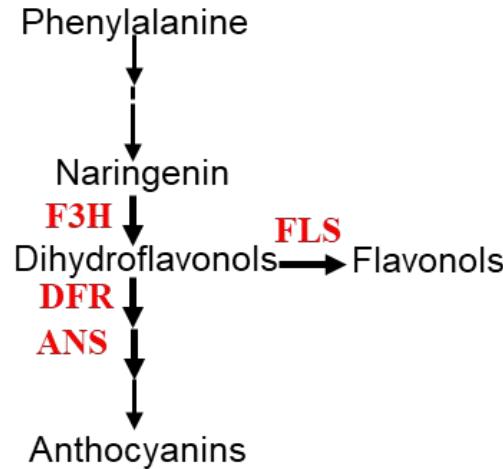


Pathway of interest



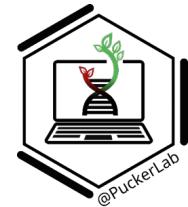


Pathway of interest



| | | Domain | Residue | |
|------------------|---|--|---------|------|
| Bait 1 | 1 | MWNIIILCIVGLVVVGITWWVYRWRNPKCRSVLPPGSMGLPLIG | | 43 |
| <u>Subject 1</u> | 3 | MWDIILCILGVVVVGITHWVYRWWRNPCKKGVLPPGSMGLPLIG | | 45 ✓ |
| <u>Subject 2</u> | 2 | MWNIIILCIVGLVVVGITWWVYKLRNPKCRSVLPPGSMGLPLIG | | 44 ✓ |
| <u>Subject 3</u> | 1 | MWNIIILCIVGLVVVGITHWVYRLSFGECKGVLPNGSMGLPLIG | | 43 ✗ |

- Confident functional assignment using conserved functional domains and residues



```
python3 /vol/data/tools/KIPEs/KIPEs3.py
```

Requirements:

dendropy 

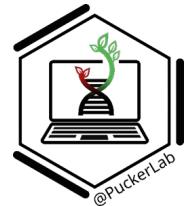
mafft 

ncbi-blast+ → sudo apt install ncbi-blast+

Subject sequences → available at </vol/data/dataset/subject/>

Example command:

```
python3 /vol/data/tools/KIPEs/KIPEs3.py --subjectdir /vol/data/dataset/subject/ --baits  
/vol/data/tools/KIPEs/baits/ --residues /vol/data/tools/KIPEs/residues/ --out  
/vol/data/user/KIPEs_results/ --cpu 13
```

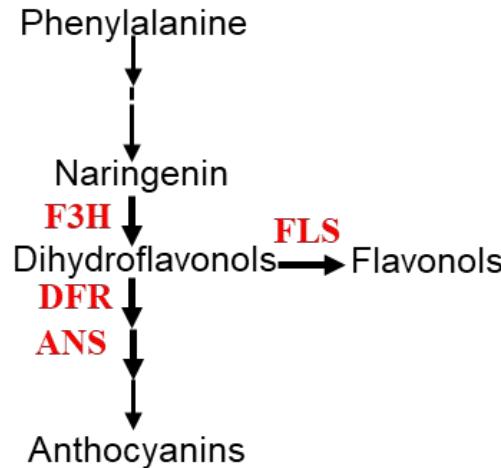
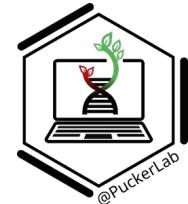


Important files/folders:

- summary.txt
- conserved_residues/
- final_pep_files/

Try :

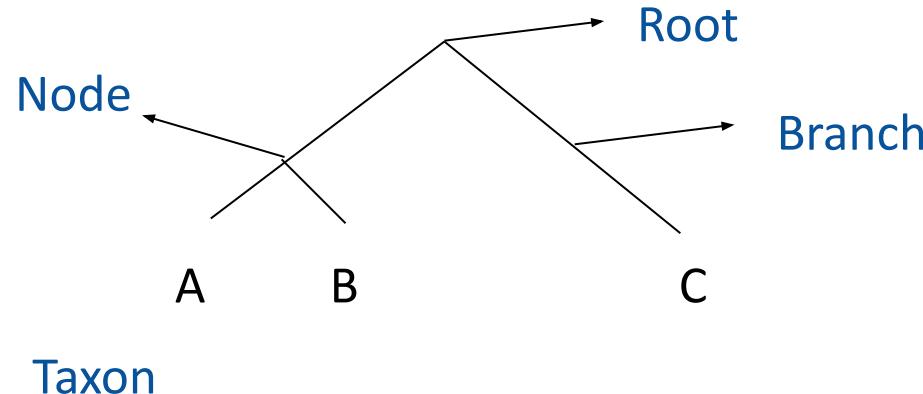
```
grep -w "100.0" summary.txt
```

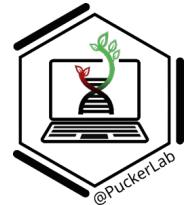


- Plant specialized metabolite pathway producing pigments
- Contains typical specialized metabolite genes like 2-ODDs, CYP450, GTs, SDRs, and so on.
- 2-ODD family members - **F3H**, **FLS**, and **ANS**



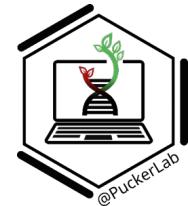
- Gene and protein sequence relationships with their ancestral sequences
- An estimate or inference - not absolute fact!





DEMO TIME!

- End to end alignment of the sequences - Global
- Alignment of most similar regions between sequences - Local



#Step 1 fetch the link from the web

```
wget https://mafft.cbrc.jp/alignment/software/mafft-7.526-linux.tgz
```

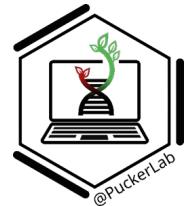
#Step 2 decompress

```
tar xfvz mafft-7.526-linux.tgz
```

#Step 3 test

```
cd mafft-linux64
```

```
execute mafft using ./mafft.bat
```



#Step 4: add mafft to the bash path (OPTIONAL)

```
nano ~/.bashrc
```

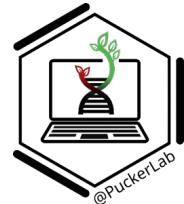
#Step 5: Add this line at the end of the bashrc file:

```
export PATH=$PATH:path/to/mafft/executable
```

#Step 6: update/ reload the bashrc file

```
source ~/.bashrc
```

Selecting the MAFFT parameters

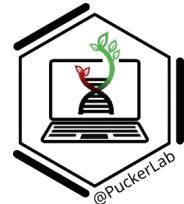


<https://mafft.cbrc.jp/alignment/software/tips0.html>

L-INS-i, E-INS-i, G-INS-i

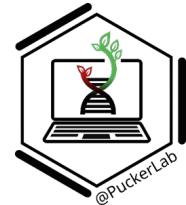
Optimal mode for our use case?

Alignment with MAFFT



#G-INS-i

```
mafft --globalpair --maxiterate 1000 --amino /vol/data/KIPEs_results_for_tree/Final_combined.pep.fasta  
> /vol/data/user/Athaliana_aligned.aln
```



Step 1: fetch the python script from GitHub

```
wget https://raw.githubusercontent.com/bpucker/ApiaceaeFNS1/main/algnttrim.py
```

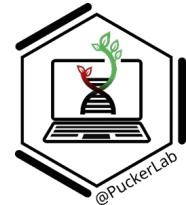
#Step 2: Understand script usage

```
python3 algnttrim.py
```

#Step 3: Run the script

```
python3 algnttrim.py --in /vol/data/user/Athaliana_aligned.aln --out  
/vol/data/user/Athaliana_aligned_trimmed.aln
```

Need to clean the alignment



- Improve the signal from the sequences for phylogeny
- Decrease noise from the alignment
- Reduces computational costs for inferring a phylogenetic tree
- Occupancy - percentage of sequences in an aligned column that don't have gaps
- We remove columns with occupancy < 10% (0.1) by default in the script (can be changed)



#Step 1: fetch the IQ-TREE3 source folder

```
cd /vol/data/tools
```

```
wget https://github.com/iqtree/iqtree3/releases/download/v3.0.1/iqtree-3.0.1-Linux.tar.gz
```

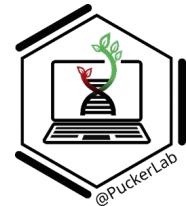
#Step 2: Decompress the tarball

```
tar -xvzf iqtree-3.0.1-Linux.tar.gz
```

```
cd iqtree-3.0.1-Linux/bin;ls
```

#Step 3: Open the bashrc file

```
nano ~/.bashrc
```



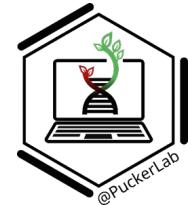
#Step 4: Add this line at the end of the bashrc file:

```
export PATH=$PATH:/vol/data/tools/iqtree-3.0.1-Linux/bin
```

#Step 5: reload the bashrc file

```
source ~/.bashrc
```

Finally tree building!



#Step 1: Look at the IQ-TREE3 tutorial pages

<https://iqtree.github.io/doc/Tutorial>

#Step 2: Run IQ-TREE3

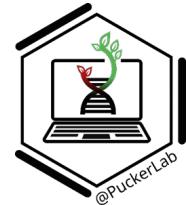
```
cd /vol/user/data
```

```
mkdir IQ-TREE3_results
```

```
iqtree3 -s /vol/data/user/Athaliana_aligned_trimmed.aln -m MFP -wsr --alrt 1000 -B 1000 -pre  
/vol/data/user/IQ-TREE3_results/2ODD -T 12
```

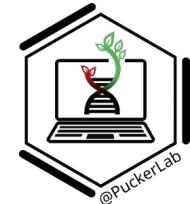


- m - MFP -> ModelFinder Plus (<https://iqtree.github.io/doc/Substitution-Models>)
- wsr -> write site rates
- alrt -> number of bootstrap replicates for SH-alrt test
- Shimodaira-Hasegawa approximate Likelihood Ratio Test -> how confident should you be about each branch in your tree
- B -> Ultrafast bootstrap
- pre -> full path to output folder with the name of your treefile

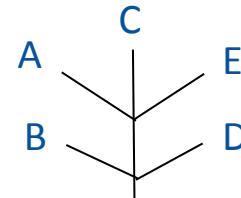


- Build an initial tree with the alignment - Original tree
- Randomly resample the alignment with replacement -> pseudo - alignment
- Obtain a tree from this pseudo - alignment
- Repeat for 1000 times (-B 1000)
- Compare original tree with the pseudo-aligned trees
- For each branch count the number of times they are exactly grouped in all the bootstrapped trees
- Get bootstrap support values for each branch

Understanding the treefile format



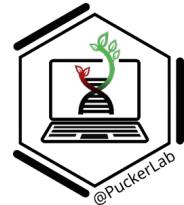
Newick standard form - represents trees in computer readable form



Newick FORMAT ->(B,(A,C,E),D);

(B:6.0,(A:5.0,C:3.0,E:4.0):5.0,D:11.0);

Trivia on trees!

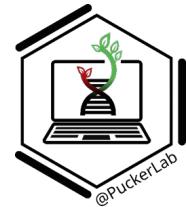


Society for Study of Evolution, New Hampshire
Finalized the treefile format at Newick's
over a plate of lobsters!

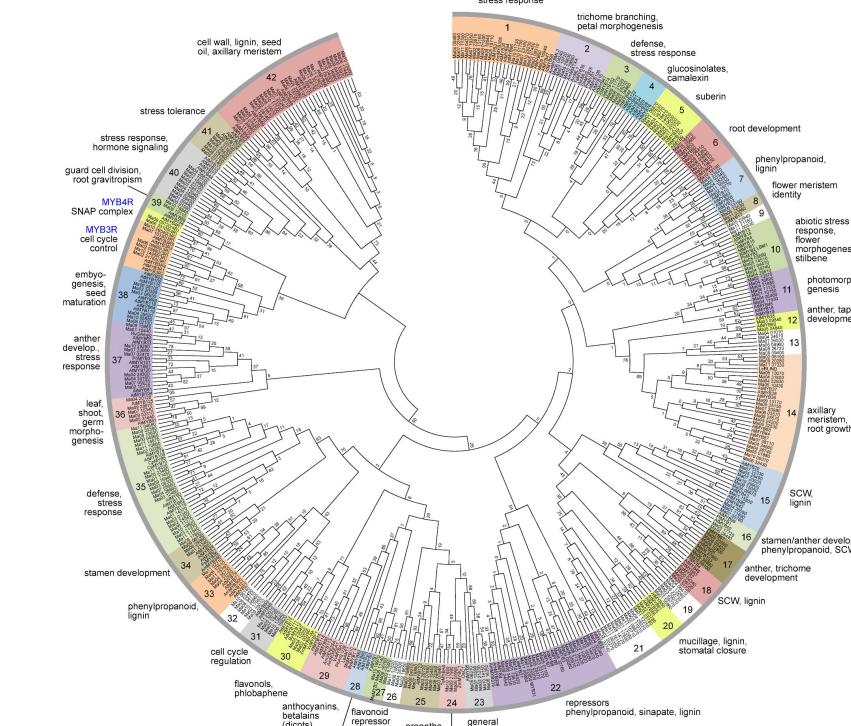


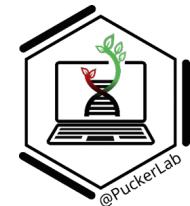
Today you learned:

- Annotating tRNAs
- Functional annotation with InterProScan5, `construct_anno.py`
- Specific functional analyses with KIPEs, `MYB_annotator`
- Multiple sequence alignment with MAFFT
- Phylogenetic tree building with IQ-TREE3

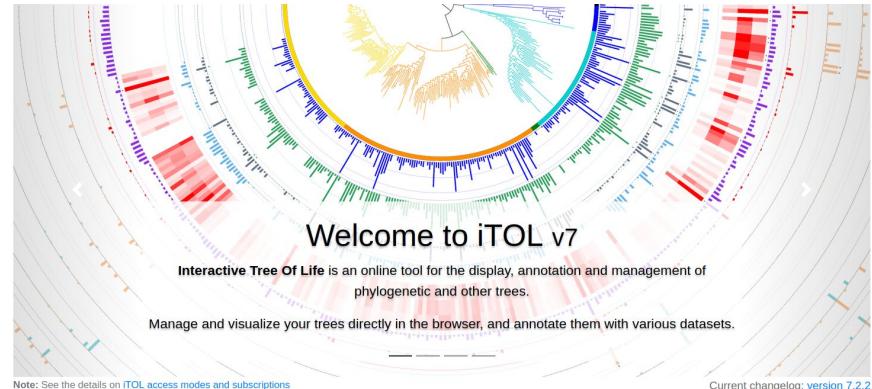


- Understanding the function of sequences based on trees
- Generation of nice illustrations for publications
- Exploration of large datasets





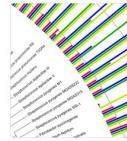
- iTOL is operated by EMBL
- Service is freely available
- Subscription is required to save modified trees (60€/year)
- Export in various file formats possible (SVG, PDF) - conversion possible





Manage

Organize your trees into workspaces and projects, and access them from any browser. Simply drag and drop multiple tree files onto a project to upload them all at once.



Annotate

19 dataset types. Full control over branch colors, widths and styles. Individually adjustable label fonts, sizes and styles. Check our [gallery of user created trees](#).

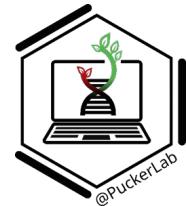


Export

Create high quality tree figures for your publications. Direct What-You-See-Is-What-You-Get export of what is displayed on the screen. Export into various vector or bitmap formats.

[Create an account](#)
[Upload a tree](#)
[Explore help](#)

Importing a tree



- Import of tree file per drag&drop possible
- Tree name is inferred from file
- Tree content can be pasted into form
- Tree name can be set manually

Sample project
Sample project. You can add more projects in the workspace options.

+ Tree upload  Settings

Upload trees into this project

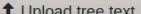
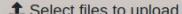
Paste the tree text or select the files to upload. You can also drag and drop the files into the box below.

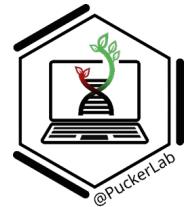
Tree text:

tree name (optional)
tree definition text

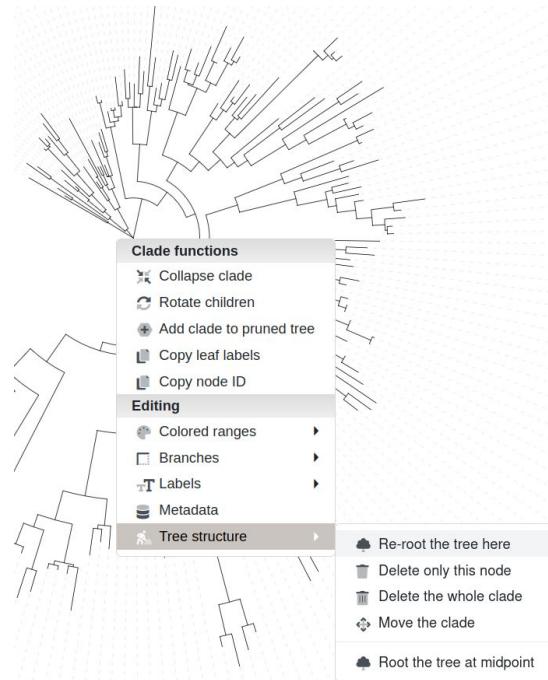
Tree files:

Drop the tree files here or



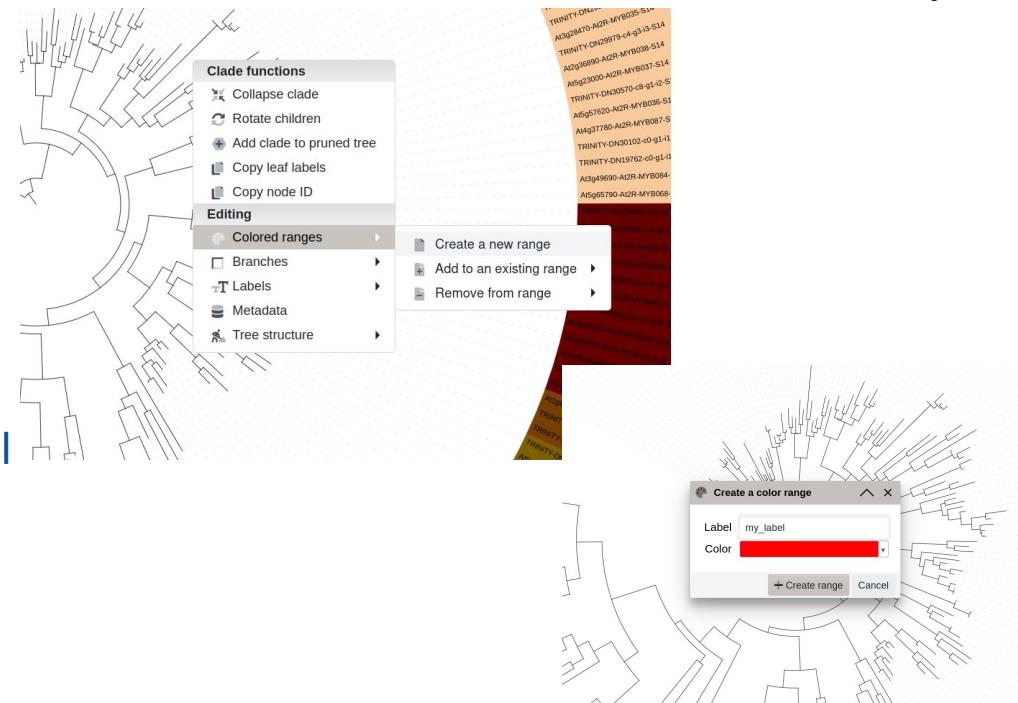
- Trees can be rooted based on an outgroup
- Left click on clade to open menu
- Go to “Tree structure”
- Select “Re-root the tree here”

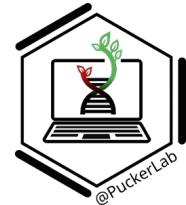


Coloring your tree in iTOL

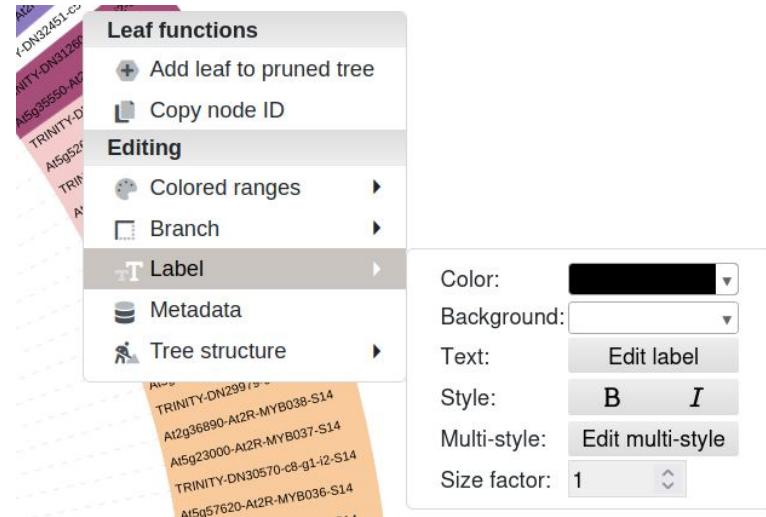


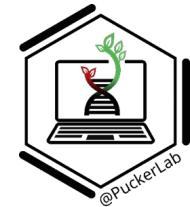
- You can highlight monophyletic clades by color
- Left click on a branch to open “Colored ranges”
- Create a new range by setting label and color



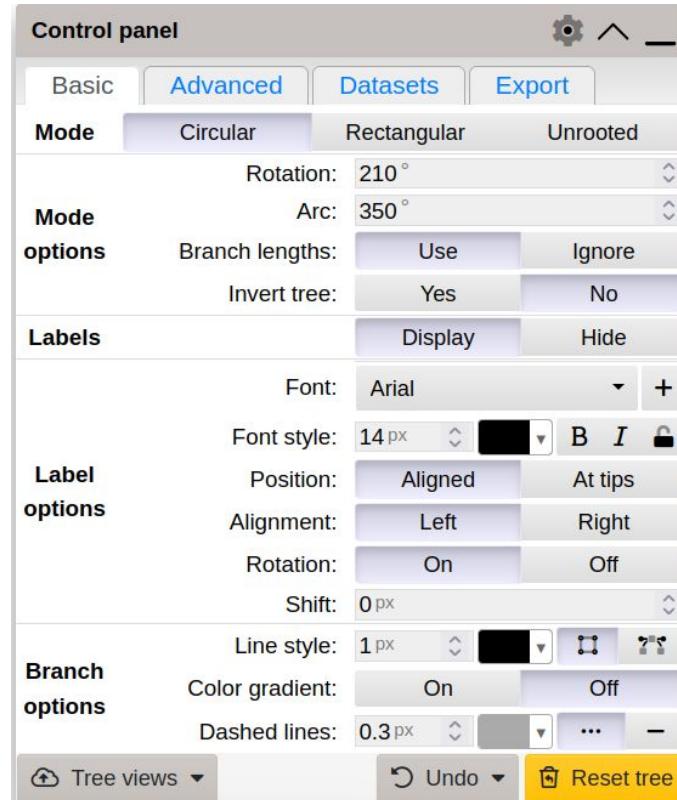


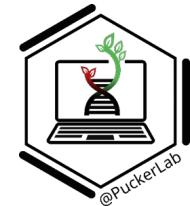
- Individual sequences can be renamed
- Font, size, style of label can be changed
- Only good option for large number of labels
(better rename in tree file)





- Mode: circular, rectangular
- Branch lengths: use/ignore
- Labels: display/hide
- Font style: px
- Position: aligned/at tips
- Line style: px





- Invert sort order: yes/no
- Node IDs: display/hide
- (Branch lengths: display/hide)
- Tree scale box: display/hide

Control panel

| | | |
|-----------------------|----------------|-----------|
| Scaling factors: | 1 x horiz. | 1 x vert. |
| Inverted circle size: | 0 px inc./dec. | |
| Leaf sorting: | Default | None |
| Invert sort order: | Yes | No |

Branch metadata display

| | | |
|-----------------|---------|------|
| Node IDs: | Display | Hide |
| Branch lengths: | Display | Hide |

Tree scales

| | | |
|----------------------|---------|------|
| Internal tree scale: | Display | Hide |
| Tree scale box: | Display | Hide |

| | |
|--|----------------|
| <input checked="" type="checkbox"/> Label font size: | 12 px |
| Label text: | Tree scale: |
| Line width: | 1 px |
| Line color: | [Color Swatch] |
| Fixed value: | 0 |

Node options

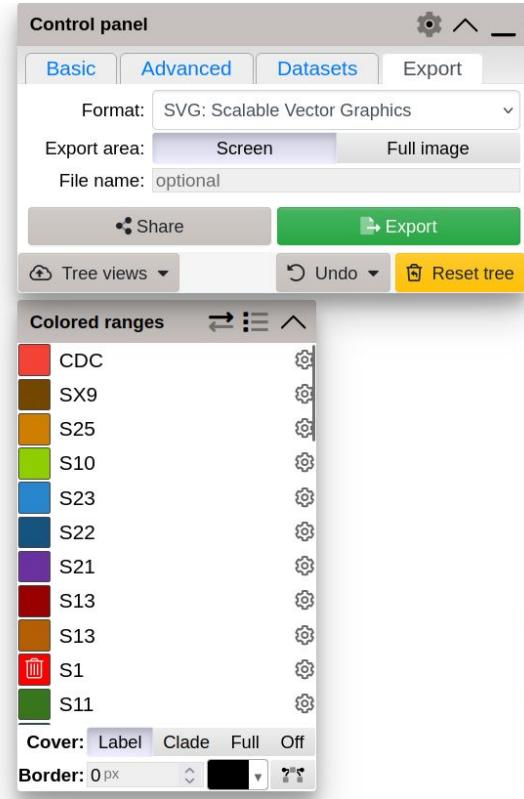
| | | |
|------------------------|-------------------|----------|
| Leaf node symbols: | Display | Hide |
| Internal node symbols: | Display | Hide |
| Auto collapse clades: | 0 > avg.BRL | Collapse |
| | no classes define | Collapse |

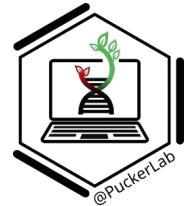
Other functions

| | | |
|-------------------------|---------------|-----------|
| Label functions: | Multi-style | Bulk edit |
| Auto assign taxonomy: | NCBI | GTDB |
| Root the tree midpoint: | Midpoint root | |

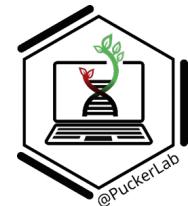


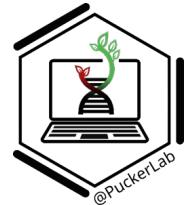
- Export as SVG, PDF, and PNG possible
- SVG/PDF can be converted on Ubuntu into any other format
- Convert function can generate almost every file type



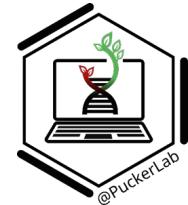


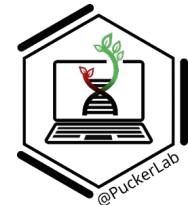
1. Import your tree
2. Root tree
3. Tree circular and rectangle mode
4. Color clades
5. Change some labels
6. Export SVG and convert into JPG



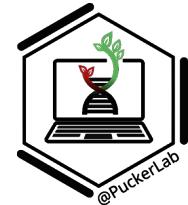


Installing DupyliCate





Availability of materials

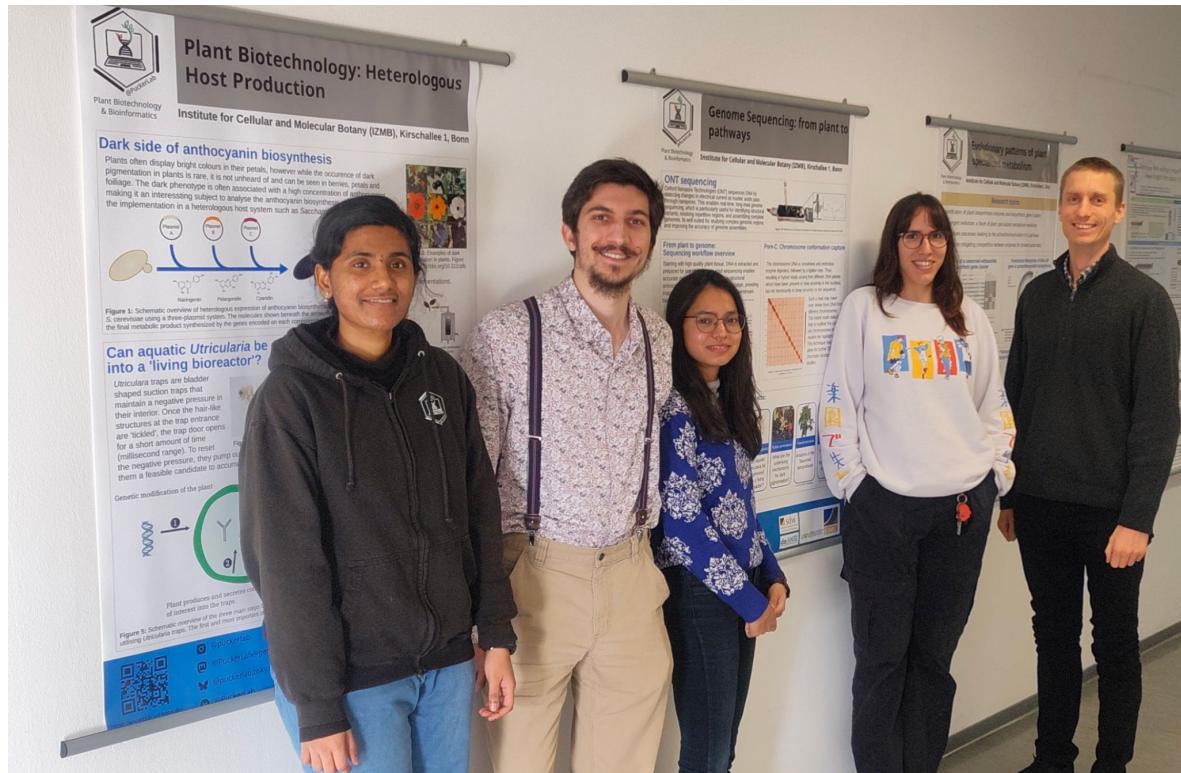


Please find all course materials here:

<https://github.com/bpucker/teaching/tree/master/deNBI2025>



Acknowledgements



From Gene Models to Biological Insights - Functional Annotation | Day 3 | 105



@PuckerLab

Email: pucker@uni-bonn.de
Web: pbb.uni-bonn.de



deNBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE