

2024 Workshop on Phylogenomics

RAxML-NG Introduction and Laboratory

Oleksiy M. Kozlov

The Exelixis Lab
Heidelberg Institute for Theoretical Studies
Germany

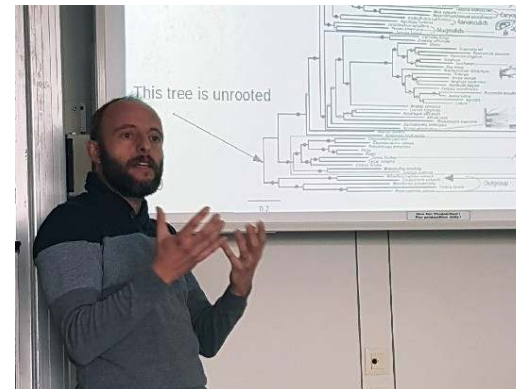
Český Krumlov – January 25, 2024

Bio

- 2007 Bachelor of Computer Science (Kyiv, Ukraine)
- 2007 – 2010 Software developer
- 2013 Master of Computer Science (KIT, Karlsruhe, Germany)
- 2013 – 2018 PhD student (Stamatakis lab)
- 2018 – ... Staff scientist (HITS, Heidelberg, Germany)
- 2023 – ... Sustainable Computing Advisor (part-time) → Sat. 2pm
- Maintainer and lead developer of RAxML-NG
- First name: Oleksiy (preferred), Alexey (before 2022)

Stamatakis lab(s)

- Prof. Alexandros Stamatakis
 - Author of the “original” RAxML
- “The Germans” → Heidelberg
 - <https://cme.h-its.org/exelixis>
 - 1 staff scientist, 1 PostDoc, 5 PhD students
- “The Cretans” → Heraklion
 - <https://www.biocomp.gr/>
 - 3 PostDocs, 2 PhD students



Outline

- RAxML-NG Intro
- Lab #1: Basics
- Results & Discussion



- RAxML-NG Parallelization
- Lab #2: Large datasets & Parallelization
- Results & Conclusions

RAxML-NG

- Maximum-likelihood (ML) phylogenetic inference tool
- Design goal #1: performance & scalability
 - B10K: 363 taxa x 63M sites ([Stiller et al., under review](#))
- Design goal #2: ease-of-use
 - Defaults & auto-tuning whenever possible
- Design goal #3: efficiency
 - Avoid wasting time & resources

Evolution of RAxML(-NG)

2006 – 2018

RAxML

(Stamatakis 2006, 2014)

- Fast & feature-rich
- Some limitations
- Hard to maintain

2018 – 2024

RAxML-NG 1.x

(Kozlov 2019)

- More flexible
- Easier to use
- Easier to maintain

2024 –

RAxML-NG 2

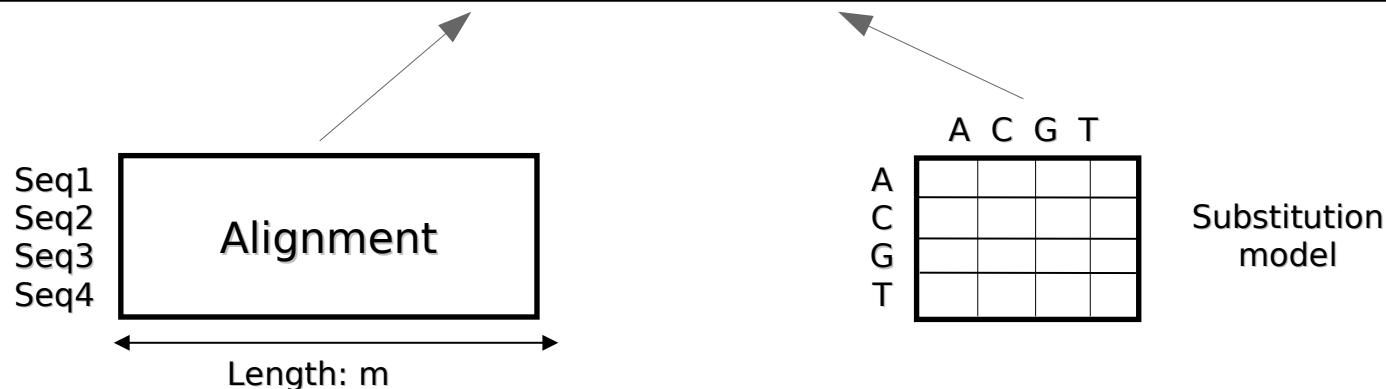
(in preparation)

- Adaptive tree search
- Fast support metrics

Quick start: ML tree search

- Default command: **--search**
 - 20 starting trees (10 random + 10 parsimony)
 - Pick the best-scoring one
 - **Disclaimer:** defaults can and will change!

```
$ raxml-ng --msa prim.phy --model GTR+G
```



Evolutionary model: Global

```
$ raxml-ng --msa prim.phy --model GTR+G
```

- Substitution matrix
 - GTR, LG, MULTI8_MK, ... → also defines input data type
- Modifiers
 - Stationary frequencies: +F0 (estimated), +FC (counted), +FE (equal)
 - Rate heterogeneity: +G (Gamma), +R (FreeRate)
 - Full list: <https://github.com/amkozlov/raxml-ng/wiki/Input-data#single-model>
- Fixed parameter values
 - HKY{1.0/2.5}+G{0.7}
 - PROTGTR{rates.txt}

Evolutionary model: Partitioned

```
$ raxml-ng --msa prim.phy --model prim.part
```

- Alignment columns divided into disjoint subsets:

```
$ cat prim.part
```

```
GTR+G+FO, NADH4=1-504  
GTR+G+FO, tRNA=505-656  
GTR+G+FO, NADH5=657-898
```

Same model, independent parameters!

```
$ cat prim2.part
```

```
GTR+G+FO, NADH4=1-504/3, 2-504/3  
JC+I, tRNA=505-656  
GTR+R4+FC, NADH5=657-898  
HKY, NADH4p3=3-504/3
```

- Branch length linkage modes:

```
--brlen linked
```

```
--brlen scaled
```

```
--brlen unlinked
```

Which model to use?

- Tools for statistical model selection
 - jModelTest, ModelFinder, SMS, ...
- We will use ModelTest-NG

```
modeltest-ng-static --help
```

Important options are:

-i ALIGNMENT

-d nt (DNA, default) or **-d aa** (proteins)



Diego Darriba

Common command line options

- Input or starting tree(s): **--tree**
 - Auto-generated: `--tree rand{5}` or `--tree pars{2},rand{2}`
 - User-specified: `--tree my.tre` or `--tree S1.raxml.bestTree`
- Output file prefix: **--prefix**
 - e.g. `S1` or `myoutput/S1` or `/home/user/S1`
 - `S1.raxml.bestTree`, `S1.raxml.log` etc
- Overwrite output files: **--redo**
- RNG seed: **--seed**
 - Deterministic search path → Reproducibility!

Tree with support values / “all-in-one”

- Command: **--all**
 - ML tree search
 - Bootstrapping (FBP) with convergence test (autoMRE)
 - Compute branch supports + map to ML tree

```
$ raxml-ng --all --msa prim.phy --model GTR+G
```

Warning: Can be extremely slow for large datasets!

(standard FBP too expensive → will be addressed in RAXML-NG 2.x)

Tree likelihood evaluation

- Command: **--evaluate**
 - Optimize all branch lengths and free model parameters
 - On a given fixed tree topology

```
$ raxml-ng --evaluate --msa prim.phy --tree S1.raxml.bestTree --model GTR+G
```

- Command: **--loglh**
 - Compute and print tree log-likelihood
 - No optimization, no files created

```
$ raxml-ng --loglh --msa prim.phy --tree S1.raxml.bestTree --model S1.raxml.bestModel
```

Comparing tree topologies

- Command: **--rfdist**
 - Compute pairwise and average Robinson-Foulds (RF) distance
 - For a given **set** of trees

```
$ raxml-ng --rfdist --tree S1.raxml.mlTrees
```

- Command: **--rf**
 - Compute and print RF distance between **two** trees

```
$ raxml-ng --rf S1.raxml.bestTree,S1A.raxml.bestTree
```

Topological constraints

- Option: **--tree-constraint**
 - Fix **some** relationships (e.g., monophyletic families)
 - By providing a (multifurcating) constraint tree
 - Can be incomplete (= some taxa “free to move”)
- Enjoy responsibly!
 - Prior knowledge vs. Bias
 - Speed vs. Bias

Outline

- RAxML-NG Intro
- [Lab #1: Basics](#)
- Results & Discussion



- RAxML-NG Parallelization
- Lab #2: Large datasets & Parallelization
- Results & Conclusions

Exercise 0: Getting ready

1. Check input datasets

```
$ cd /home/phylogenomics/workshop_materials/  
$ git clone https://github.com/amkozlov/ng-tutorial  
$ cd ng-tutorial  
$ ls
```

2. Run raxml-ng without parameters to get help

```
$ raxml-ng
```

3. Check alignment for formatting errors → [prim.phy](#)

```
$ raxml-ng --check --msa prim.phy --model GTR+G
```

Exercises: Lab #1

1. Tree search
2. Bootstrapping / “all-in-one”
3. Tree likelihood evaluation
4. Partitioned models
5. Topological constraints
6. Model testing

<https://github.com/amkozlov/ng-tutorial/blob/master/evomics2024.md>

Outline

- RAxML-NG Intro
- Lab #1: Basics
- Results & Discussion



- RAxML-NG Parallelization
- Lab #2: Large datasets & Parallelization
- Results & Conclusions

Exercise 1: Answers

2. ML tree likelihoods

```
$ grep "logLikelihood:" S1.raxml.log
```

```
[00:00:00] [worker #0] ML tree search #1, logLikelihood: -5708.940514
[00:00:00] [worker #2] ML tree search #3, logLikelihood: -5709.367652
[00:00:00] [worker #3] ML tree search #4, logLikelihood: -5708.950769
[00:00:00] [worker #1] ML tree search #2, logLikelihood: -5708.981882
[00:00:00] [worker #0] ML tree search #5, logLikelihood: -5708.969503
[00:00:00] [worker #2] ML tree search #7, logLikelihood: -5708.949393
[00:00:00] [worker #1] ML tree search #6, logLikelihood: -5708.936930
[00:00:00] [worker #3] ML tree search #8, logLikelihood: -5709.023648
[00:00:01] [worker #0] ML tree search #9, logLikelihood: -5708.976056
[00:00:01] [worker #2] ML tree search #11, logLikelihood: -5709.009527
[00:00:01] [worker #1] ML tree search #10, logLikelihood: -5708.943575
[00:00:01] [worker #3] ML tree search #12, logLikelihood: -5709.015055
[00:00:01] [worker #0] ML tree search #13, logLikelihood: -5708.968725
[00:00:01] [worker #2] ML tree search #15, logLikelihood: -5709.020541
[00:00:01] [worker #1] ML tree search #14, logLikelihood: -5709.013903
[00:00:01] [worker #3] ML tree search #16, logLikelihood: -5709.012914
[00:00:01] [worker #0] ML tree search #17, logLikelihood: -5709.010325
[00:00:01] [worker #2] ML tree search #19, logLikelihood: -5709.021621
[00:00:01] [worker #1] ML tree search #18, logLikelihood: -5709.021075
[00:00:01] [worker #3] ML tree search #20, logLikelihood: -5709.006120
```

Exercise 1: Answers (2)

3. Average topological (RF) distance

```
$ raxml-ng --rfdist --tree S1.raxml.mlTrees --prefix RF1
```

```
Reading input trees from file: S1.raxml.mlTrees  
Loaded 20 trees with 12 taxa.
```

```
Average absolute RF distance in this tree set: 0.000000  
Average relative RF distance in this tree set: 0.000000  
Number of unique topologies in this tree set: 1
```

Absolute RF = # branches **not shared** by both trees

Relative RF = Absolute RF / max. possible RF

Exercise 1: Answers (3)

4*. fusob.phy

```
$ raxml-ng --msa fusob.phy --model GTR+G --prefix S2 --tree pars{5},rand{5}
```

```
$ grep "logLikelihood:" S2.raxml.log
```

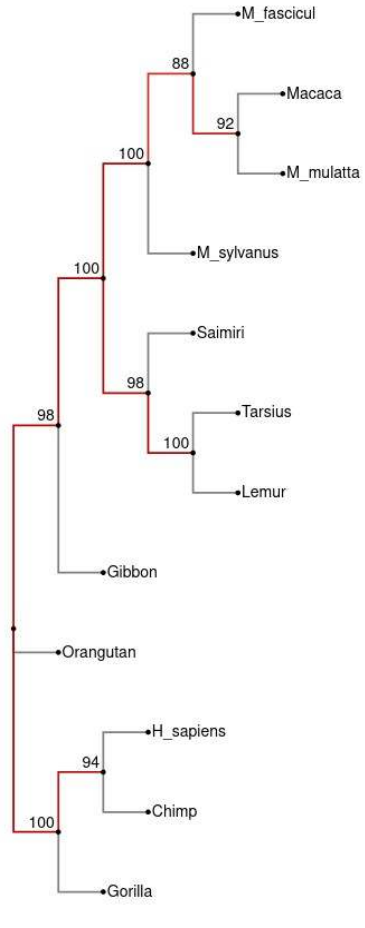
```
[00:00:02] [worker #2] ML tree search #3, logLikelihood: -9974.673091
[00:00:02] [worker #0] ML tree search #1, logLikelihood: -9974.665558
[00:00:02] [worker #1] ML tree search #2, logLikelihood: -9980.906638
[00:00:03] [worker #3] ML tree search #4, logLikelihood: -9974.663320
[00:00:04] [worker #2] ML tree search #7, logLikelihood: -9980.906494
[00:00:04] [worker #1] ML tree search #6, logLikelihood: -9974.670122
[00:00:04] [worker #3] ML tree search #8, logLikelihood: -9974.669716
[00:00:05] [worker #0] ML tree search #5, logLikelihood: -9974.671526
[00:00:06] [worker #1] ML tree search #10, logLikelihood: -9974.666947
[00:00:07] [worker #0] ML tree search #9, logLikelihood: -9981.909653
```

```
$ raxml-ng --rfdist --tree S2.raxml.mLTrees --prefix RF2
```

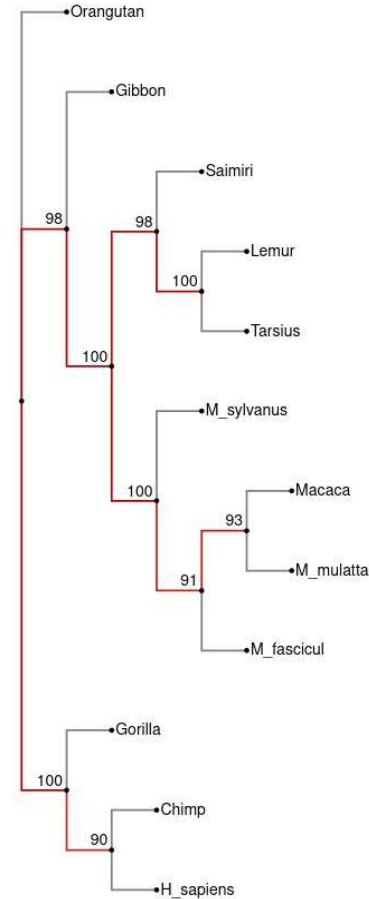
```
Average absolute RF distance in this tree set: 5.244444
Average relative RF distance in this tree set: 0.074921
Number of unique topologies in this tree set: 3
```

Exercise 2: Answers

50 reps



100 reps



Exercise 3: Answers

```
$ grep "Final LogLikelihood:" E*.raxml.log
```

```
E_GTRG.raxml.log:Final LogLikelihood: -5715.693753
```

```
E_GTRR4.raxml.log:Final LogLikelihood: -5714.610468
```

Best: GTR+R

```
E_GTR.raxml.log:Final LogLikelihood: -5934.558984
```

```
E_JCG.raxml.log:Final LogLikelihood: -6272.478819
```

```
E_JC.raxml.log:Final LogLikelihood: -6424.202453
```

```
$ grep "AIC score" E*.raxml.log
```

Best: GTR+G

```
E_GTRG.raxml.log:AIC score: 11491.387506 / AICc score: 11493.532834 / BIC score: 11635.392608
```

```
E_GTRR4.raxml.log:AIC score: 11499.220936 / AICc score: 11502.144370 / BIC score: 11667.226889
```

```
E_GTR.raxml.log:AIC score: 11927.117968 / AICc score: 11929.122576 / BIC score: 12066.322900
```

```
E_JCG.raxml.log:AIC score: 12588.957638 / AICc score: 12590.114210 / BIC score: 12694.561380
```

```
E_JC.raxml.log:AIC score: 12890.404907 / AICc score: 12891.459701 / BIC score: 12991.208478
```

Exercise 4: Answers

```
$ grep "Final LogLikelihood:" {S,P}1.raxml.log
```

```
S1.raxml.log:Final LogLikelihood: -5708.926872
```

```
P1.raxml.log:Final LogLikelihood: -5673.806570
```

```
$ raxml-ng --rf S1.raxml.bestTree,P1.raxml.bestTree
```

```
Average absolute RF distance in this tree set: 0.000000
```

```
Average relative RF distance in this tree set: 0.000000
```

```
Number of unique topologies in this tree set: 1
```

Exercise 5: Answers

```
$ grep "Final LogLikelihood:" C*.raxml.log
```

C0.raxml.log:Final LogLikelihood: -5708.923405	no constraint
C1.raxml.log:Final LogLikelihood: -5708.926239	constraint #1
C2.raxml.log:Final LogLikelihood: -5779.491583	constraint #2

```
$ grep "Elapsed time:" C*.raxml.log
```

C0.raxml.log:Elapsed time: 23.629 seconds	no constraint
C1.raxml.log:Elapsed time: 17.553 seconds	constraint #1
C2.raxml.log:Elapsed time: 17.987 seconds	constraint #2

Exercise 6: Answers

```
$ modeltest-ng -i prot21.fa -d aa
```

Partition 1/1:

	Model	Score	Weight

BIC	LG+G4	6005.4554	0.5062
AIC	LG+I+G4	5893.6825	0.7923
AICc	LG+G4	5941.3599	0.5402

```
$ raxml-ng --msa prot21.fa --model LG+G4 --prefix S6
```

Final LogLikelihood: -2872.979205

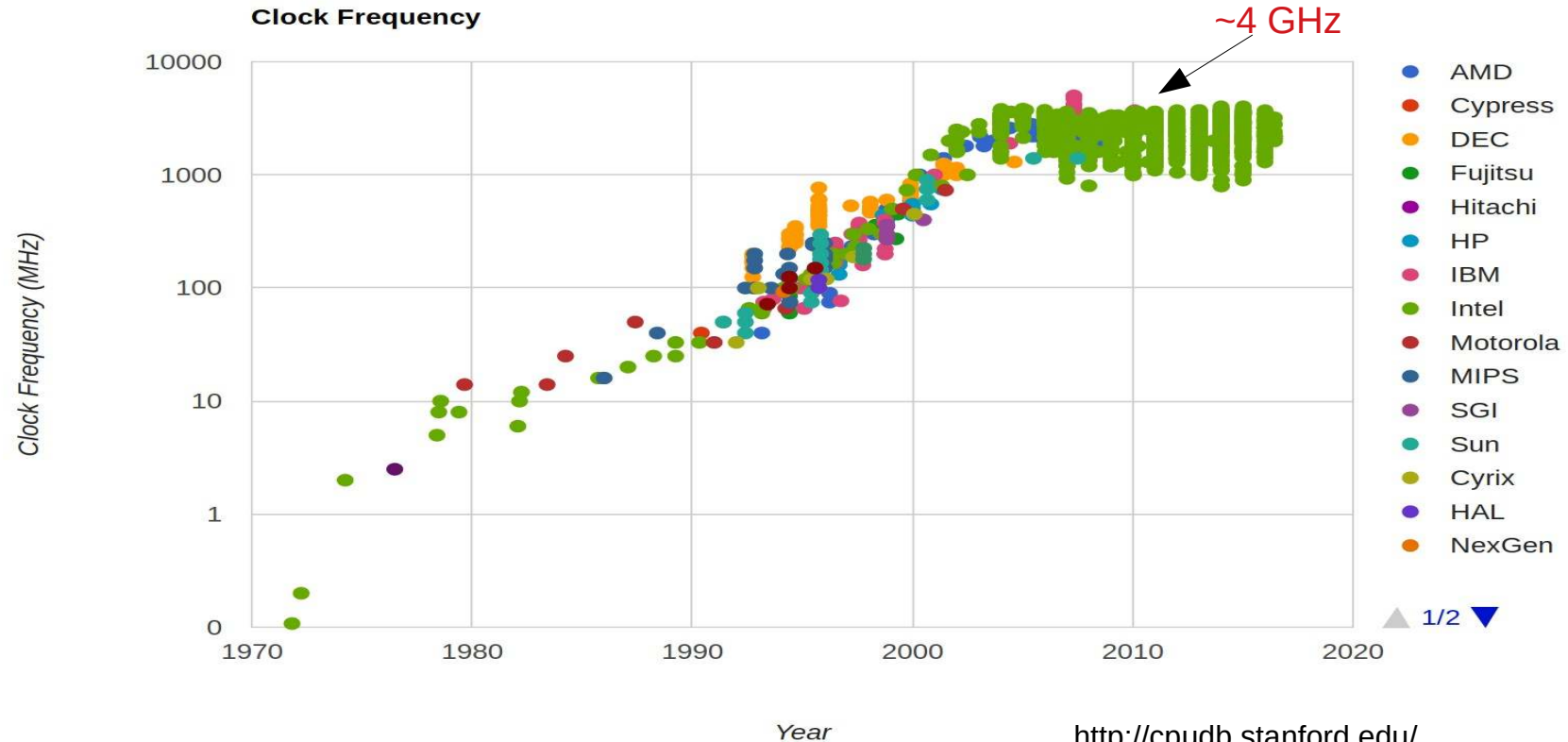
Outline

- RAxML-NG Intro
- Lab #1: Basics
- Results & Discussion

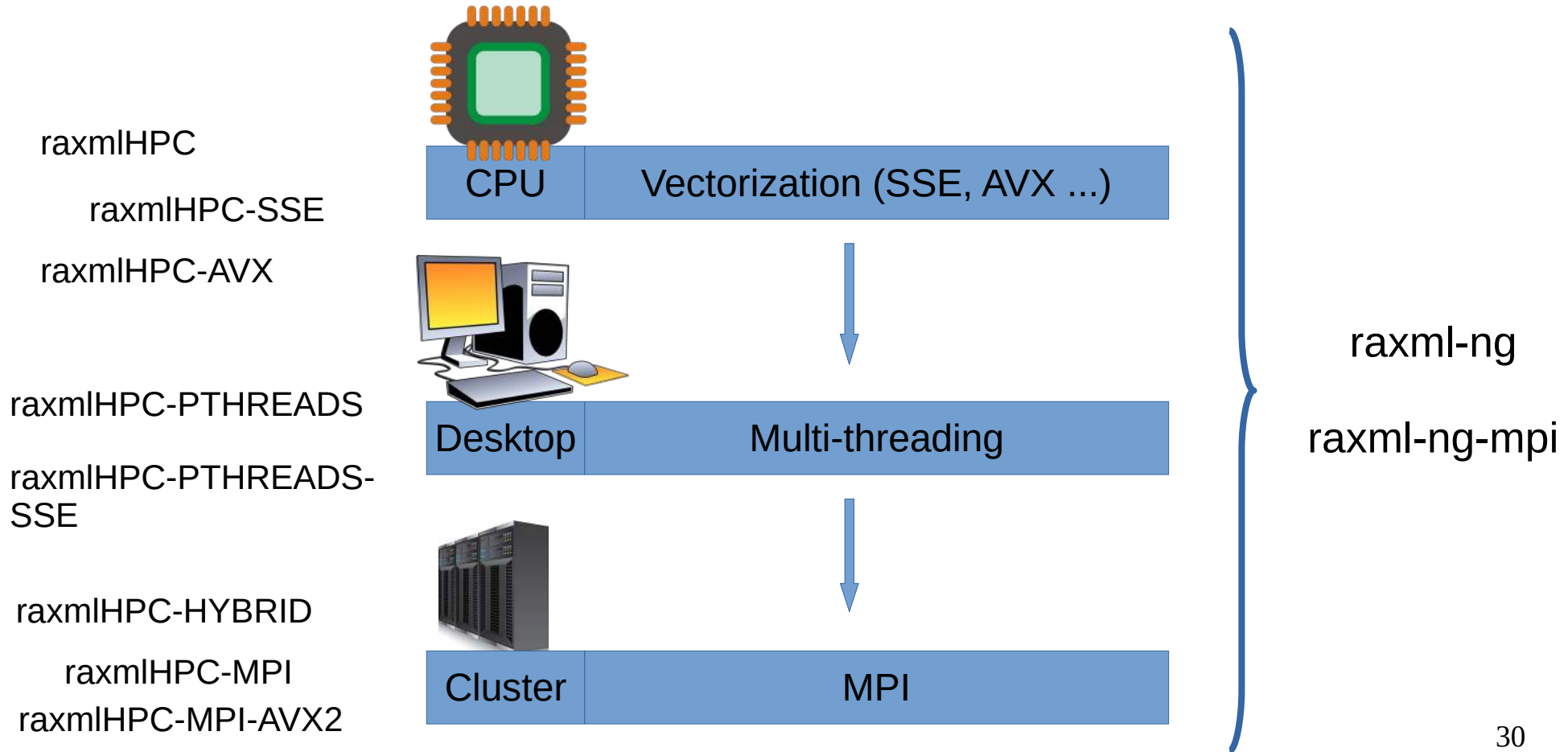


- [RAxML-NG Parallelization](#)
- Lab #2: Large datasets & Parallelization
- Results & Conclusions

Why is parallelization so important?



Parallelization: hardware

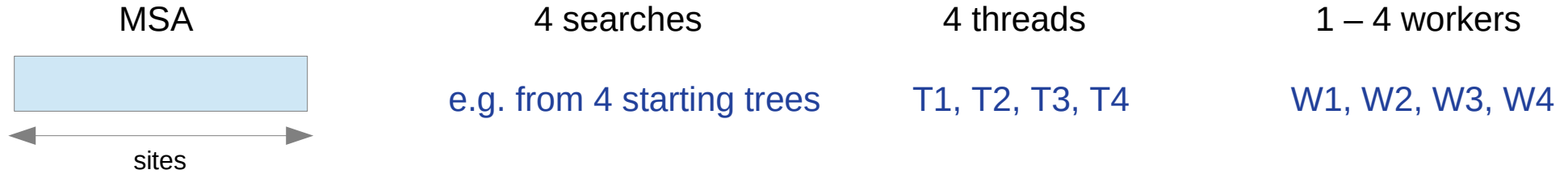


RAXML-NG parallelization setup

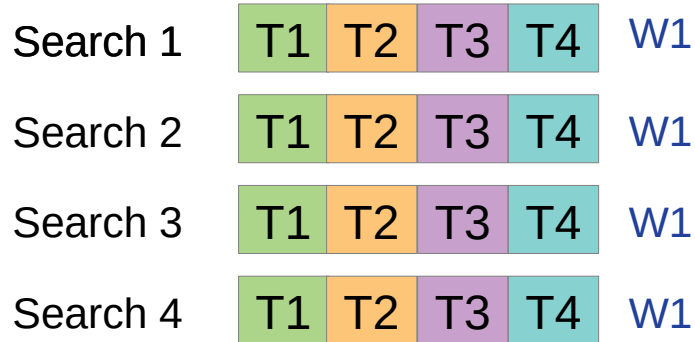
- **Vectorization** → fully automatic
- **Multi-threading** → usually works fine™
 - Automatic CPU detection & thread tuning
 - Core oversubscription warning
- **MPI/hybrid** → more tricky
 - Read your cluster manual
 - Ask your sysadmin/technician
 - Benchmark!



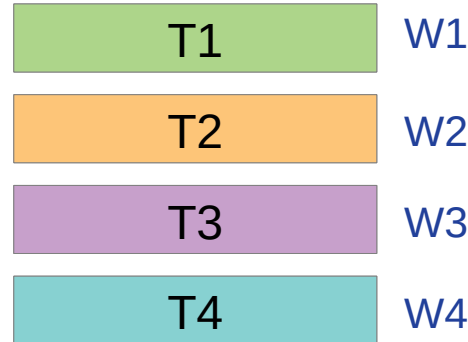
Parallelization: software



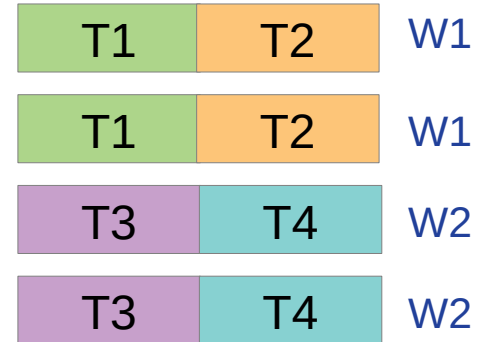
Fine-grained



Coarse-grained



Mixed/hybrid



Since v1.0: Full native support and automatic configuration!

Parallelization: auto-tuning

```
$ raxml-ng --msa prim.rba
```

```
System: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, 16 cores, 62 GB RAM
```

```
...
```

```
Analysis options:
```

```
  run mode: ML tree search
```

```
  start tree(s): random (10) + parsimony (10) ←
```

```
...
```

```
  parallelization: coarse-grained (auto), PTHREADS (auto)
```

```
...
```

```
[00:00:00] Alignment comprises 12 taxa, 1 partitions and 413 patterns
```

```
...
```

```
Parallelization scheme autoconfig: 16 worker(s) x 1 thread(s) ←
```

```
...
```

```
[00:00:00] Data distribution: max. partitions/sites/weight per thread: 1 / 413 / 6608
```

```
[00:00:00] Data distribution: max. searches per worker: 2 ←
```

Parallelization: manual tuning

- Automatic with upper limits

```
$ raxml-ng --msa prim.rba --threads auto{16} --workers auto{2}
```

- Manual

```
$ raxml-ng --msa prim.rba --threads 16 --workers 2
```

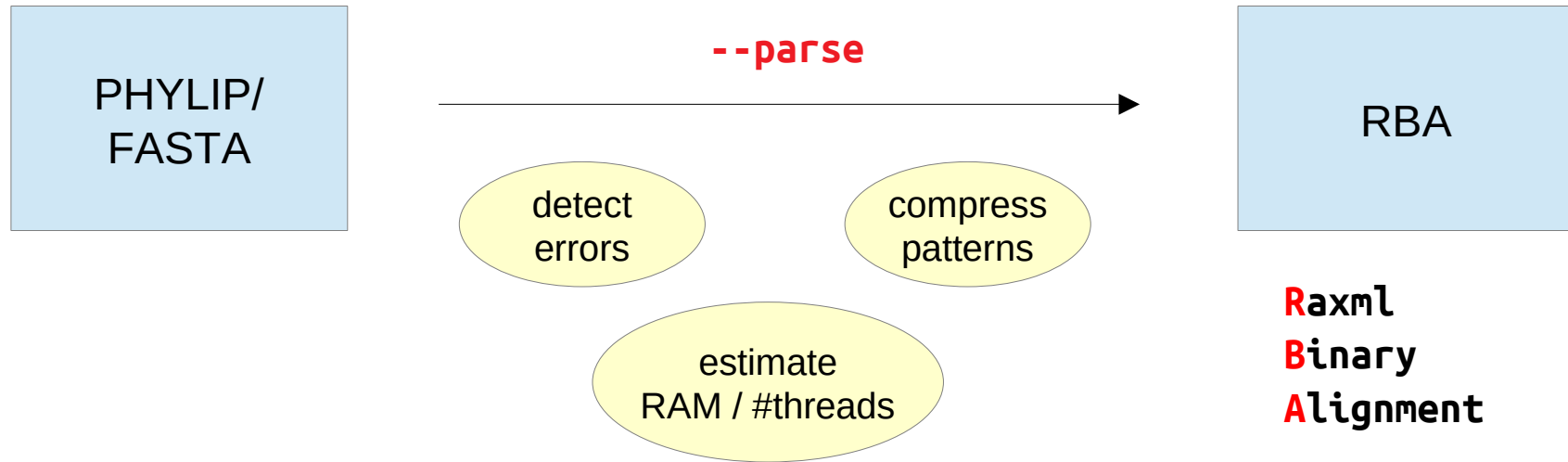
- Also works with MPI

```
$ mpirun -n 4 raxml-ng-mpi --msa prim.rba --threads 16 --workers 8
```

4 ranks * 16 threads = 64 = 8 workers * 8 threads

Alignment pre-processing

```
$ raxml-ng --parse --msa prim.phy --model GTR+G --prefix prim
```



Bootstrapping revisited

1. Infer bootstrap trees

```
$ raxml-ng --bootstrap --msa prim.phy --model GTR+G --prefix BS1 --bs-trees 50 --seed $RANDOM
```

2. Check convergence

```
$ raxml-ng --bsconverge --bs-trees BS1.raxml.bootstraps --prefix BC
```

3. Map bootstrap support values to the best ML tree

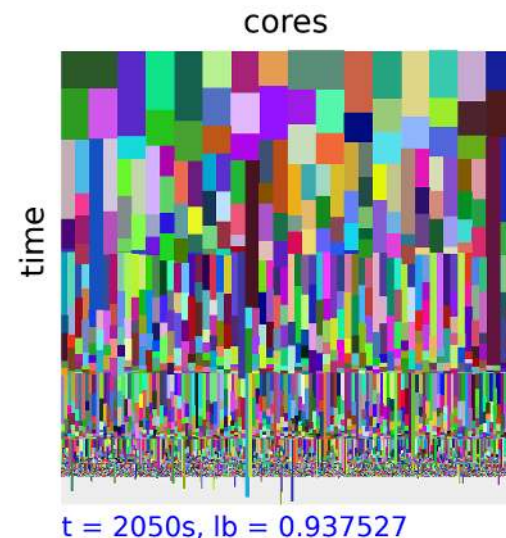
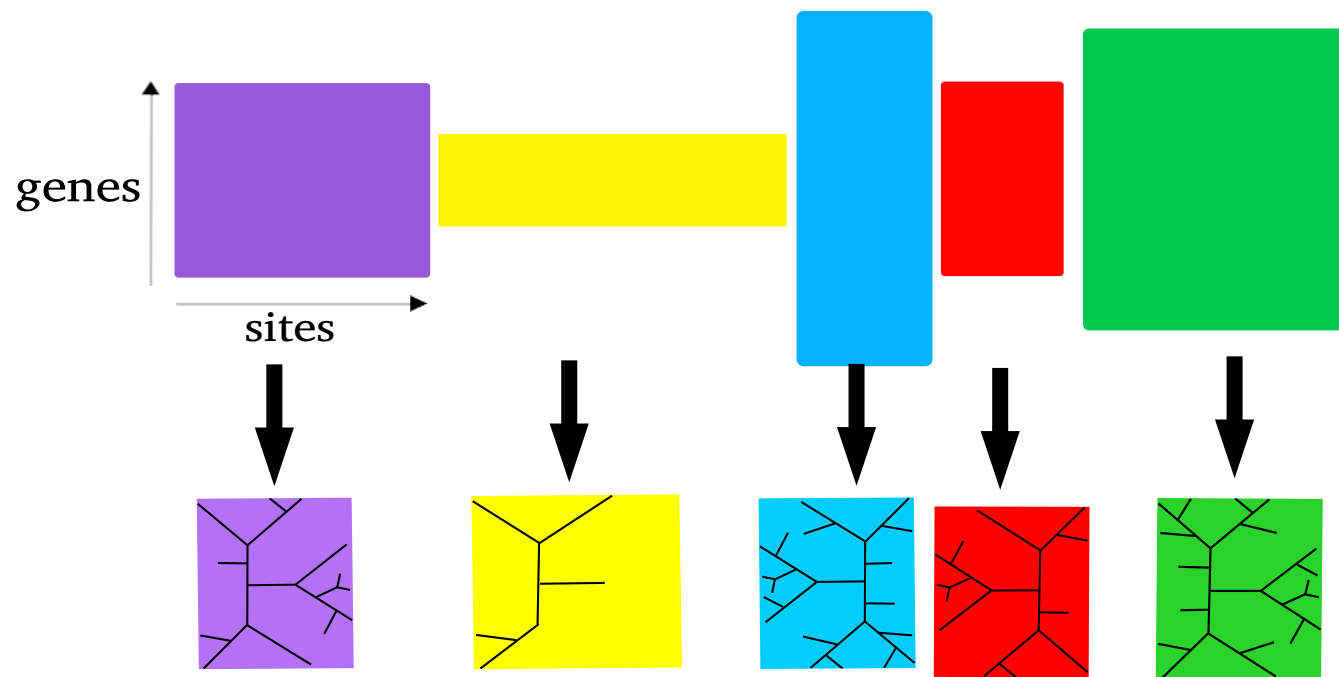
```
$ raxml-ng --support --tree S1.raxml.bestTree --bs-trees BS1.raxml.bootstraps --prefix B2
```

ParGenes

- Infer thousands of (gene) trees in parallel
 - Load balancing + Checkpointing
 - Model testing (ModelTest-NG) + species tree reconstruction (ASTRAL)



(Morel 2018)



ParGenes command line - example

Do not run this one!

Folder containing
the alignments
-a msa_dir

Folder for
result files
-o output_dir

Number of
cores
-c 256

```
$ python pargenes.py -a msa_dir -o output_dir -m -s 10 -p 20 -b 100 -p 256
```

Apply model
selection?
-m

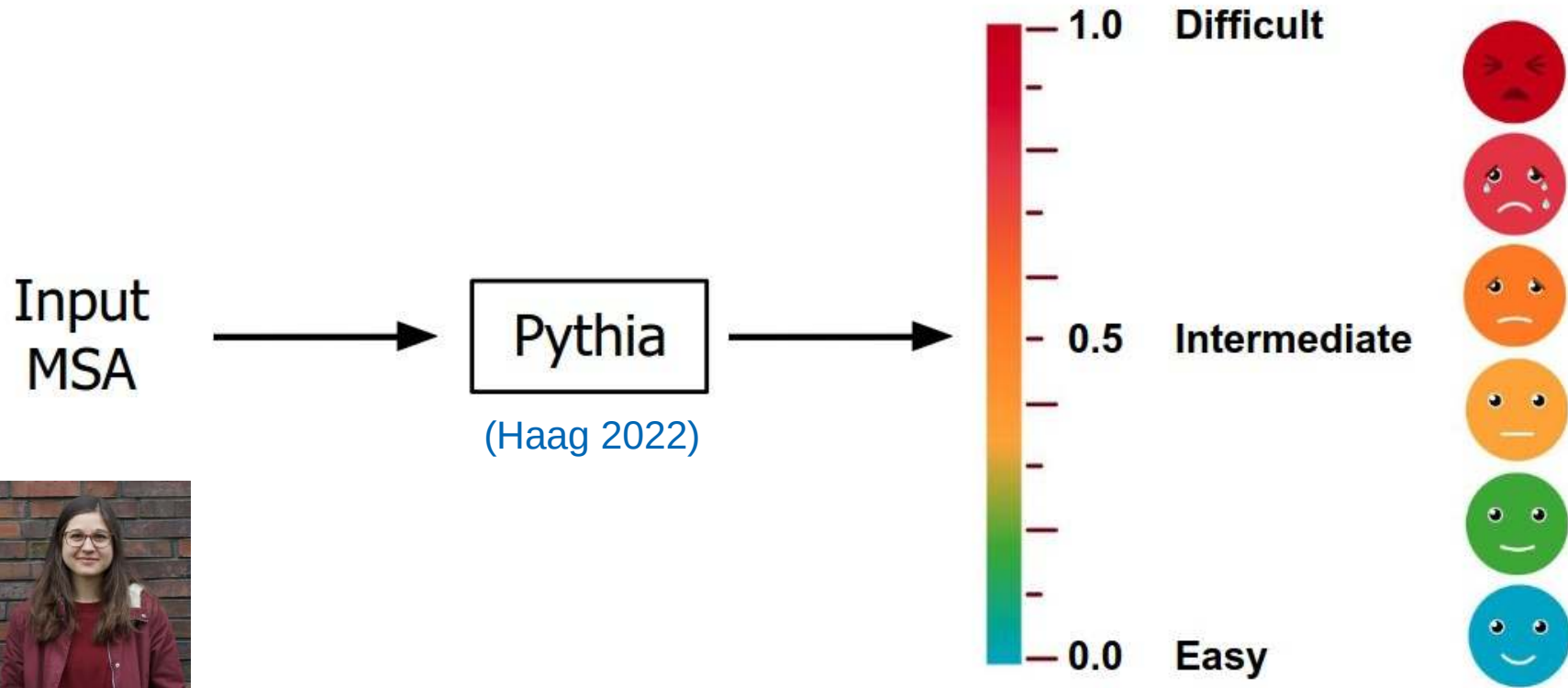
Number of
starting trees
-p 20 -s 10

Number of BS
replicates
-b 100

More examples: </home/phylogenomics/software/.source/ParGenes/examples/>

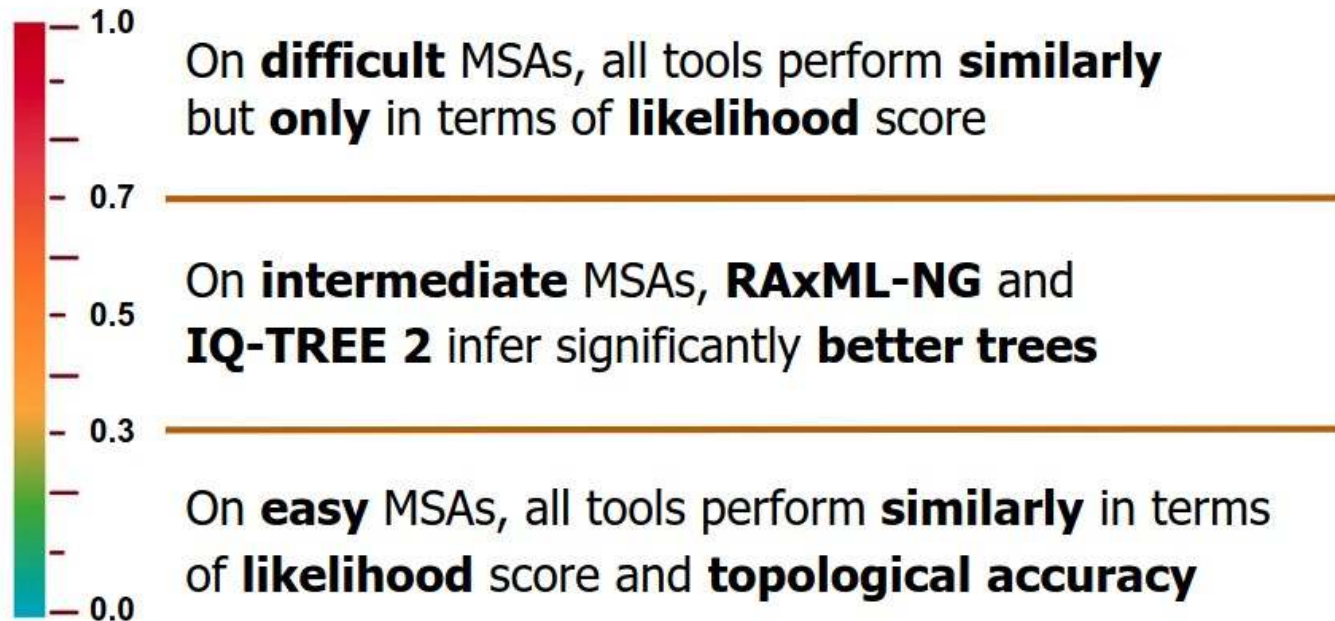
“From easy to hopeless”

Prediction of phylogenetic difficulty



Julia Haag

- RAxML-NG vs. IQ-TREE 2 vs. FastTree [\(Hoehler, 2022 preprint\)](#)

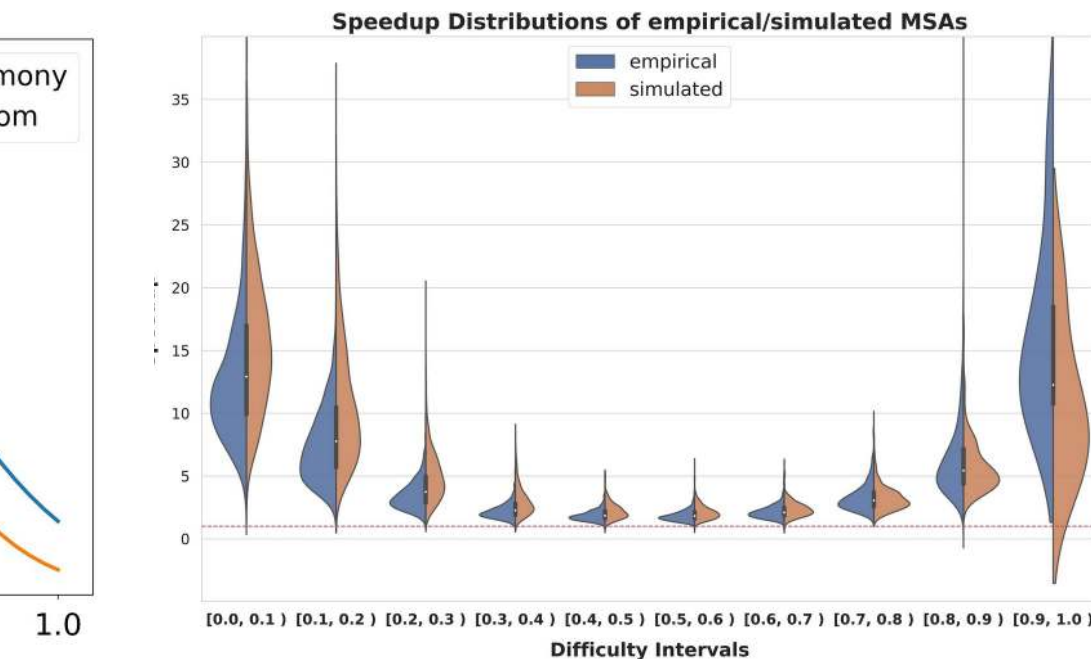
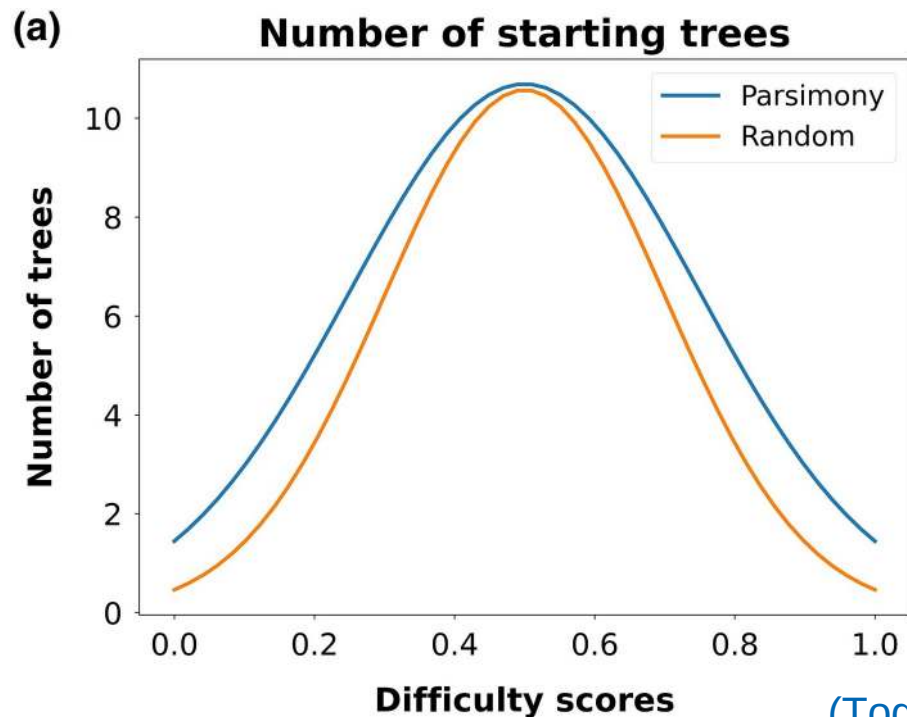


Adaptive RAxML-NG

- Use predicted MSA difficulty to adjust
- number of starting trees + search heuristic (SPR radius, NNIs)



Anastasis Togkousidis



(Togkousidis 2023)

Outline

- RAxML-NG Intro
- Lab #1: Basics
- Results & Discussion



- RAxML-NG Parallelization
- [Lab #2: Large datasets & Parallelization](#)
- Results & Conclusions

Knowing your system

```
$ lscpu
```

```
CPU(s):          4  logical
Thread(s) per core: 2  hyperthreading ON
Model name:      AMD EPYC 7571
```

```
$ raxml-ng -v
```

```
System: AMD EPYC 7571, 2 cores, 15 GB RAM
physical
```

```
$ htop
```

```
1[|||||          9.2%]  Tasks: 160, 556 thr; 1 running
2[|||||          4.5%]  Load average: 0.16 0.18 0.13
3[              0.0%]  Uptime: 08:55:25
4[|||||          12.3%]
Mem[|||||        2.27G/15.5G]
Swp[              0K/0K]
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	163M	12112	8272	S	0.0	0.1	0:05.47	/sbin/init
134	root	19	-1	48576	19840	18560	S	0.0	0.1	0:01.39	/lib/systemd/systemd-journald

Exercises: Lab #2

- 7. Alignment compression
- 8. Automatic & manual parallelization
- 9. Bootstrapping revisited
- 10. Adaptive search
- 11. ParGenes

<https://github.com/amkozlov/ng-tutorial/blob/master/evomics2024.md>

Outline

- RAxML-NG Intro
- Lab #1: Basics
- Results & Discussion



- RAxML-NG Parallelization
- Lab #2: Large datasets & Parallelization
- Results & Conclusions

Exercise 7: Answers

```
$ raxml-ng --parse --msa fusob.phy --model GTR+G --prefix fusob
```

```
Partition 0: noname
```

```
Model: GTR+FO+G4m
```

```
Alignment sites / patterns: 1602 / 635
```

```
Gaps: 10.13 %
```

```
Invariant sites: 9.61 %
```

```
NOTE: Binary MSA file created: fusob.raxml.rba
```

```
* Estimated memory requirements           : 6 MB
```

```
* Recommended number of threads / MPI processes: 3
```

Exercise 8: Answers

```
raxml-ng -search -msa fusob.raxml.rba -tree rand{10} -seed 1 -prefix TXWX  
raxml-ng -search -msa fusob.raxml.rba -tree rand{10} -seed 1 -workers 1 -prefix TXW1  
raxml-ng -search -msa fusob.raxml.rba -tree rand{10} -seed 1 -workers 2 -prefix TXW2  
raxml-ng -search -msa fusob.raxml.rba -tree rand{10} -seed 1 -threads 2 -workers 2 -prefix T2W2
```

```
$ grep "Elapsed time:" T?W?.raxml.log
```

```
T2W2.raxml.log:Elapsed time: 11.995 seconds
```

```
TXW1.raxml.log:Elapsed time: 9.752 seconds
```

```
TXW2.raxml.log:Elapsed time: 8.551 seconds
```

```
TXWX.raxml.log:Elapsed time: 10.398 seconds
```



~18 % faster

Exercise 9: Answers

```
$ raxml-ng --bootstrap --msa prim.phy --model GTR+G --prefix BS1 --seed $RANDOM --bs-trees 50  
$ raxml-ng --bootstrap --msa prim.phy --model GTR+G --prefix BS2 --seed $RANDOM --bs-trees 50
```

```
$ cat BS*.raxml.bootstraps > bstrees.txt
```

```
$ raxml-ng --support -tree S1.raxml.bestTree -bs-trees bstrees.txt -prefix B2
```

```
$ raxml-ng --support -tree S1.raxml.bestTree -bs-trees bstrees.txt -prefix B3 -bs-metric TBE
```

Exercise 10: Answers

Analysis options:

run mode: ML tree search

Starting ML tree search
with 20 distinct starting trees

Final LogLikelihood: -5673.760245

Elapsed time: 8.618 seconds

Analysis options:

run mode: Adaptive ML tree search

[00:00:00] Predicted difficulty: 0.00

Starting ML tree search
with 2 distinct starting trees

Final LogLikelihood: -5673.844106

Elapsed time: 1.635 seconds

```
$ raxml-ng-adaptive --rf P1.raxml.bestTree,P1A.raxml.bestTree
```

Average absolute RF distance in this tree set: 0.000000

Average relative RF distance in this tree set: 0.000000

Number of unique topologies in this tree set: 1

Exercise 11: Answers

```
$ python ~/software/ParGenes/pargenes/pargenes.py  
-a ~/software/ParGenes/examples/data/small/fastq_files/ -o parout  
-c 2 -m --scheduler openmp
```

Logs will be redirected to parout2/parse_run/logs.txt

Average number of taxa: 21

Max number of taxa: 21

Average number of sites: 111

Max number of sites: 111

Recommended MAXIMUM number of cores: 3

[0:00:11] end of the second parsing step

[0:00:22] end of mlsearch mpi-scheduler run

[0:00:23] end of selecting the best ML tree

[0:00:23] END OF THE RUN OF pargenes.py

Software availability

- Web server
 - CIPRES: <http://www.phylo.org/> → registration required
- Graphical interface
 - raxmlGUI: <https://github.com/AntonelliLab/raxmlGUI>
 - Mesquite: <http://www.mesquiteproject.org/>
- Stand-alone command line (Linux/macOS)
 - GitHub: <https://github.com/amkozlov/raxml-ng>
 - Conda: <https://anaconda.org/bioconda/raxml-ng>

Where to get help?

- Documentation
 - <https://github.com/amkozlov/raxml-ng/wiki>
- Tutorial
 - <https://github.com/amkozlov/raxml-ng/wiki/Tutorial>
- User support group
 - <https://groups.google.com/forum/#!forum/raxml>

Which tool to use as of 2024 ?

- FastTree
 - “Hopeless” alignments (100k+, few genes)
- RAxML, IQ-Tree, PhyML
 - Models/features not available in RAxML-NG
- ParGenes
 - Lots of gene trees, coalescent methods
- RAxML-NG
 - All other cases :)

Děkuji

Questions?

References

- Barbera et al. (2018) **EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences.** *Systematic Biology*, syy054, <https://doi.org/10.1093/sysbio/syy054>
- Kozlov et al. (2018) **RAXML-NG: A fast, scalable, and user-friendly tool for maximum likelihood phylogenetic inference.** *bioRxiv*. <https://doi.org/10.1101/447110>
- Kozlov, Aberer, Stamatakis (2015) **ExaML version 3: a tool for phylogenomic analyses on supercomputers.** *Bioinformatics*. <https://doi.org/10.1093/bioinformatics/btv184>
- Morel, Kozlov, Stamatakis (2018) **ParGenes: a tool for massively parallel model selection and phylogenetic tree inference on thousands of genes.** *Bioinformatics*. <https://doi.org/10.1093/bioinformatics/bty839>
- Stamatakis (2006) **RAXML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models.** *Bioinformatics*, <https://doi.org/10.1093/bioinformatics/btl446>
- Stamatakis (2014) **RAXML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies.** *Bioinformatics*, <https://doi.org/10.1093/bioinformatics/btu033>
- Stamatakis and Aberer (2013) **Novel parallelization schemes for large-scale likelihood-based phylogenetic inference.** *In Parallel Distributed Processing (IPDPS)* <https://doi.org/10.1109/IPDPS.2013.70>