2019 Workshop on Phylogenomics

RAxML-NG Introduction and Laboratory

Alexey M. Kozlov

The Exelixis Lab
Heidelberg Institute for Theoretical Studies
Germany

Český Krumlov – January 24, 2019
Outline

- RAxML-NG Intro
- Lab1: Basics
- RAxML-NG Parallelization
- Lab2: Parallelization
- Conclusions

Cheatsheet (all commands and results):
https://github.com/amkozlov/ng-tutorial/wiki/evomics2019
Evolution of RAxML

- 2003: v1.0
- 2006: v3.0
- 2013: v8.0
- 2015: v8.2.12
- 2017: v3.0.21
- 2020: v0.1

- 2019: v0.8

- ExaML
- RAxML-NG
- RAxML
- RAxML-Light
Why RAxML-NG?

- RAxML is cool
  - Tons of features
  - Fast
  - >20k citations
- RAxML is ugly
  - 77k lines of legacy C code
  - Maintenance nightmare
RAxML-NG vs. RAxML

• Complete re-write
  – Search heuristic is the same
  – Numerous bugfixes & optimizations

• Benefits for us
  – Easier to maintain & extend

• Benefits for you
  – Easier to use
  – Even faster
  – More flexible & reliable
RAxML-NG family

libpll / pll-modules

- Vectorized PLF kernels (SSE3, AVX, AVX2)
- Numerical optimization
- Discrete tree operations
- File I/O (MSA & trees)
- MSA validation & statistics
- Parsimony

other apps

RAxML
(Stamatakis '06, '14)

RAxML-NG
(Kozlov '18)

ModelTest-NG
(Darriba in prep.)

EPA-NG
(Barbera '18)

ParGenes
(Morel '18)

(Flouri in prep.)
Outline

- RAxML-NG Intro
- **Lab1: Basics**

- RAxML-NG Parallelization
- Lab2: Parallelization
- Conclusions
Exercise 0: Getting ready

1. Check input datasets

   $ cd /home/phylogenomics/workshop_materials/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

4*. Run check for bad.fa & examine error messages
Common command line options

```
$ raxml-ng --msa prim.phy --model GTR+G --prefix S1
```

- **Default command:** *--search*
  - 10 random + 10 parsimony starting trees
  - *--tree*, e.g. *--tree rand{5} or --tree pars{2},rand{2}*
  - *--search1* is a shortcut for *--search --tree rand{1}*

- **Evolutionary model:** *--model*
  - Single model (*GTR+G*) or partition file (*mypart.txt*)

- **Output file prefix:** *--prefix*
  - e.g. *S1* or *myoutput/S1* or */home/user/S1*
  - *S1.raxml.bestTree, S1.raxml.log* etc.
Exercise 1: Tree search

1. Run tree search for \texttt{prim.phy} with default parameters

\begin{verbatim}
$ raxml-ng --msa prim.phy --model GTR+G --prefix S1
\end{verbatim}

2. Compare likelihoods of all 20 resulting trees
   - Hint: use \texttt{grep} command on \texttt{S1.raxml.log} file!

3. Check topological distances between all 20 trees (so-called Robinson-Foulds or RF distance)

\begin{verbatim}
$ raxml-ng --rfdist --tree S1.raxml.mlTrees --prefix RF1
\end{verbatim}

4*. Repeat step 1-3 for \texttt{fusob.phy}
   - use 3 parsimony + 3 random starting trees
Exercise 1: Answers

2. ML tree likelihoods

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

```plaintext
[00:00:00] ML tree search #1, logLikelihood: -5708.961164
[00:00:01] ML tree search #2, logLikelihood: -5709.001321
[00:00:02] ML tree search #3, logLikelihood: -5708.928444
[00:00:03] ML tree search #4, logLikelihood: -5708.958315
[00:00:03] ML tree search #5, logLikelihood: -5708.932260
[00:00:04] ML tree search #6, logLikelihood: -5708.941449
[00:00:05] ML tree search #7, logLikelihood: -5708.959505
[00:00:05] ML tree search #8, logLikelihood: -5708.951658
[00:00:06] ML tree search #9, logLikelihood: -5709.022061
[00:00:07] ML tree search #10, logLikelihood: -5708.926872
[00:00:08] ML tree search #11, logLikelihood: -5709.016549
[00:00:08] ML tree search #12, logLikelihood: -5709.022648
[00:00:09] ML tree search #13, logLikelihood: -5709.009746
[00:00:10] ML tree search #14, logLikelihood: -5709.012081
[00:00:10] ML tree search #15, logLikelihood: -5709.017948
[00:00:11] ML tree search #16, logLikelihood: -5709.017067
[00:00:11] ML tree search #17, logLikelihood: -5709.030238
[00:00:12] ML tree search #18, logLikelihood: -5709.014300
[00:00:13] ML tree search #19, logLikelihood: -5709.018029
[00:00:13] ML tree search #20, logLikelihood: -5709.072513
```
Exercise 1: Answers (2)

3. Average topological (RF) distance

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{3},rand{3}

$ grep "logLikelihood:" S2.raxml.log

[00:00:07] ML tree search #1, logLikelihood: -9974.666846
[00:00:13] ML tree search #2, logLikelihood: -9974.669424
[00:00:20] ML tree search #3, logLikelihood: -9974.673880
[00:00:25] ML tree search #4, logLikelihood: -9980.602445
[00:00:30] ML tree search #5, logLikelihood: -9974.670042
[00:00:36] ML tree search #6, logLikelihood: -9980.601596

$ raxml-ng --rfdist --tree S2.raxml.mlTrees --prefix RF2

Reading input trees from file: S2.raxml.mlTrees
Loaded 6 trees with 38 taxa.

Average absolute RF distance in this tree set: 4.266667
Average relative RF distance in this tree set: 0.060952
Number of unique topologies in this tree set: 2
Exercise 2: Bootstrapping

1. Run bootstrap tree inference with default parameters

```
$ raxml-ng --bootstrap --msa prim.phy --model GTR+G --prefix B1
```

2. Map bootstrap support values to the best ML tree

```
$ raxml-ng --support --tree S1.raxml.bestTree --bs-trees B1.raxml.bootstraps --prefix B2
```

3. Open the resulting tree with support values in the tree viewer of your choice.

4*. Repeat bootstrapping using a fixed number of replicates (100). Did this changed the support values?
Exercise 2: Answers
Combined search & bootstrapping

• Command: **--all**

```bash
$ raxml-ng --all --msa prim.phy --model GTR+G --prefix A1
```

• Convenient for small datasets
Exercise 3: Tree likelihood evaluation

• Command: **--evaluate**
  
  – By default, re-optimizes all branch lengths and model parameters

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

1. Evaluate the likelihood of `S1.raxml.bestTree` under the following models: **GTR+G, GTR+R4, GTR, JC** and **JC+G**. Don't forget to change the **--prefix**!

2. Compare likelihoods and AIC/AICc/BIC scores *(lower=better)*. Which model should be preferred and why?
Exercise 3: Answers

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

E_GTR.raxml.log:Final LogLikelihood: -5934.159081
E_GTRG.raxml.log:Final LogLikelihood: -5709.005399
**E_GTRR.raxml.log:Final LogLikelihood: -5706.012286**
E_JC.raxml.log:Final LogLikelihood: -6424.203377
E_JCG.raxml.log:Final LogLikelihood: -6272.469065

Best: GTR+R

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

E_GTR.raxml.log:AIC score: 11926.318162 / AICc score: 11928.322770 / BIC score: 12065.523094
**E_GTRG.raxml.log:AIC score: 11478.010798 / AICc score: 11480.156127 / BIC score: 11622.015900**
E_GTRR.raxml.log:AIC score: 11482.024572 / AICc score: 11484.948006 / BIC score: 11650.030524
E_JC.raxml.log:AIC score: 12890.406755 / AICc score: 12891.461549 / BIC score: 12991.210326
E_JCG.raxml.log:AIC score: 12588.938129 / AICc score: 12590.094701 / BIC score: 12694.541871

Best: GTR+G
Exercise 4: Protein data & ModelTest-NG

1. Check online help

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

Important options are:

- `-i ALIGNMENT`
- `-d nt` (DNA, default) or `-d aa` (proteins)

2. Run model selection for `prot21.fa` (protein alignment!)

3. Run tree inference with the best-scoring model determined by ModelTest-NG
Exercise 4: Answers

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

Partition 1/1:

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIC</td>
<td>LG+G4</td>
<td>6005.4554</td>
</tr>
<tr>
<td>AIC</td>
<td>LG+I+G4</td>
<td>5893.6825</td>
</tr>
<tr>
<td>AICc</td>
<td>LG+G4</td>
<td>5941.3599</td>
</tr>
</tbody>
</table>

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

Final LogLikelihood: -2872.979205
Exercise 5: Partitioned models

• Partitioned model definition

$ cat prim2.part

GTR+G+F0, NADH4=1-504/3,2-504/3
JC+I, tRNA=505-656
GTR+R4+FC, NADH5=657-898
HKY, NADH4p3=3-504/3

1. Re-run tree inference for prim.phy using partitioned model in prim2.part

2. Compare the results (log-likelihood and tree topology) to the Exercise 1.
Exercise 5: Answers

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

S1.raxml.log:Final LogLikelihood: -5708.926872
P1.raxml.log:Final LogLikelihood: -5673.806570
```

```
$ cat S1.raxml.bestTree P1.raxml.bestTree > S1P1.trees

$ raxml-ng --rfdist --tree S1P1.trees --prefix RF5

Reading input trees from file: S1P1.trees
Loaded 2 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
```
Exercise 6: RAxML-NG Web Server

- Please visit: https://raxml-ng.vital-it.ch/
- Play around (e.g., repeat some Exercises)
Outline

● RAxML-NG Intro
● Lab1: Basics

● RAxML-NG Parallelization
● Lab2: Parallelization
● Conclusions
Why is parallelization so important?

~4 GHz

http://cpudb.stanford.edu/
Moore’s law vs. Brooks’ law

diminishing returns
Levels of parallelism

raxmlHPC
raxmlHPC-SSE
raxmlHPC-AVX
raxmlHPC-PTHREADS
raxmlHPC-PTHREADS-SSE
raxmlHPC-HYBRID
raxmlHPC-MPI
raxmlHPC-MPI-AVX2

CPU
Vectorization (SSE, AVX ...)

Desktop
Multi-threading

Cluster
MPI

raxml-ng
raxml-ng-mpi
RAxML-NG parallelization setup

• **Vectorization** → fully automatic 😊

• **Multi-threading** → needs some attention
  - Hardware: 1 thread per **physical** core! 😐
  - Dataset: use **--parse** to get recommendation

• **MPI/hybrid** → more tricky
  - Read your cluster manual 😒
  - Ask your sysadmin/technician
  - Benchmark!
RAxML-NG parallelization modes

**Fine-grained**

<table>
<thead>
<tr>
<th>Search 1</th>
<th>Search 2</th>
<th>Search 3</th>
<th>Search 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
</tbody>
</table>

**Coarse-grained**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
</tbody>
</table>

**Mixed/hybrid**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
</tbody>
</table>

MSA (Multiple Sequence Alignment) with 4 threads and 4 searches, e.g., from 4 starting trees.

- **Natively supported by RAxML-NG**
- **Custom scripts or ParGenes**
ParGenes

- Infer thousands of (gene) trees in parallel
  - Load balancing
  - Checkpointing
  - Integrated model testing (ModelTest-NG)

(Morel 2018)
Outline

- RAxML-NG Intro
- Lab1: Basics
- RAxML-NG Parallelization
- Lab2: Parallelization
- Conclusions
Knowing your system

$ lscpu

CPU(s):              2
Thread(s) per core:  1
Model name:          Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
hyperthreading disabled

$ htop

Tasks: 87, 185 thr: 1 running
Load average: 0.33 0.25 0.20
Uptime: 5 days, 03:44:00

PID USER   PRI NI VIRT  RES  SHR S   CPU% MEM% TIME+ Command
430 phylogeno 20  0 647M 135M  62136 S 10.7 1.7  8h33:37 /usr/lib/xorg/Xorg :10 -auth .Xauthority -config xrdp/xorg.conf -norecovery
3079 phylogeno 20  0  364M 29260 21608 S  1.3  0.4  0:05.15 /usr/lib/mate-screensaver/mate-screensaver/floaters /usr/share/pixmaps
575 phylogeno 20  0 1046M 37832 28380 S  0.7  0.5  28:07.99 marco
3625 phylogeno 20  0  34040  6344  3732 R  0.7  0.1  0:00.96 htop
### Hyperthreading

```bash
$ lscpu

CPU(s):                      8
On-line CPU(s) list:        0-7
Thread(s) per core:         2
Core(s) per socket:         4
Socket(s):                   1
NUMA node(s):               1
Vendor ID:                   GenuineIntel
CPU family:                 6
Model:                      142
Model name:                 Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz

1 | | |     2.6% | 5 | | |     2.6%
2 | | |     2.6% | 6 | | |     0.7%
3 | | |     2.6% | 7 | | |     2.6%
4 | | |     0.7% | 8 | | |     3.9%
Mem |          | 10.3G/23.3G | | | | | Task: 196, 912 thr; 1 running
                  | 312K/2.00G   |

Load average: 0.52 0.38 0.33
Uptime: 31 days, 13:39:23
```
Exercise 7: Alignment compression

• Command: --parse
  - Compress alignment patterns
  - Generate binary alignment file → *.rba
  - Estimate resource consumption (memory, # threads)

1. Compress alignment file fusob.phy

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

2*. Explore how resource estimates change depending on the selected --model (e.g., GTR, GTR+R8)
Exercise 7: Answers

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: Fine-grained parallelization

1. Run the same tree search with 1 and then with 2 threads

   $ raxml-ng --search --msa fusob.raxml.rba --tree rand{10} --seed 1 --threads 1 --prefix T1

   $ raxml-ng --search --msa fusob.raxml.rba --tree rand{10} --seed 1 --threads 2 --prefix T2

2. Compare runtimes. Which number of threads is “optimal”?

3*. Try to oversubscribe CPU cores by using 3 or 4 threads. What do you observe?
Exercise 8: Answers

$ grep "Elapsed time:" T*.raxml.log

T1.raxml.log:Elapsed time: 68.191 seconds
T2.raxml.log:Elapsed time: 48.026 seconds

$ raxml-ng --search --msa fusob.raxml.rba --tree rand{10} --seed 1 --threads 3 --prefix T3

parallelization: PTHREADS (3 threads), thread pinning: OFF

[00:00:00 -18709.360432] Initial branch length optimization
[00:00:30 -16006.630258] Model parameter optimization (eps = 10.000000)
[00:01:53 -14327.870042] AUTODETECT spr round 1 (radius: 5)
Exercise 9: Coarse-grained parallelization

1. Start two RAxML-NG instances in the background:
   - type this command on one single line!
   - Do not forget the ampersand (&)

   ```bash
   for i in {1..2}; do (raxml-ng --msa fusob.raxml.rba --tree rand{5} --seed $i --threads 1 --prefix CT$i >CTlog$i & ); done
   ```

2. Use `htop` program to monitor progress
   - look at per-core CPU load!

3. Compare runtimes with fine-grained parallelization (Ex. 8)
Exercise 9: Answers

$ grep "Elapsed time:" CT*.raxml.log T*.raxml.log

CT1.raxml.log:Elapsed time: 40.091 seconds
CT2.raxml.log:Elapsed time: 37.830 seconds
T1.raxml.log:Elapsed time: 68.191 seconds
T2.raxml.log:Elapsed time: 48.026 seconds

~20 % faster

$ grep "Final LogLikelihood" CT*.raxml.log | sort -k 3

CT1.raxml.log:Final LogLikelihood: -9974.663429
CT2.raxml.log:Final LogLikelihood: -9974.663779
Exercise 10: ParGenes

$ python ~/software/ParGenes/pargenes/pargenes.py --help

-a ALIGNMENTS_DIR, --alignments-dir ALIGNMENTS_DIR
   Directory containing the fasta files
-o OUTPUT_DIR, --output-dir OUTPUT_DIR
   Output directory
-c CORES, --cores CORES
   The number of computational cores available for computation. Should at least 2.
--msa-filter MSA_FILTER
   A file containing the names of the msa files to process
-d {nt,aa}, --datatype {nt,aa}
   Alignments datatype
--scheduler {split,onecore,openmp}
   Scheduling strategy. Prefer split for multiple nodes platforms, and openmp else (for instance when running on your personal computer.
-s RANDOM_STARTING_TREES, --random-starting-trees RANDOM_STARTING_TREES
   The number of starting trees
-p PARSIMONY_STARTING_TREES, --parsimony-starting-trees PARSIMONY_STARTING_TREES
   The number of starting parsimony trees
-m, --use-modeltest
   Autodetect the model with modeltest
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

Apply model selection?
- m

Number of starting trees
- p 20 - s 10

Number of BS replicates
- b 100

More examples: /home/phylogenomics/software/ParGenes/examples/
Exercise 10: ParGenes

1. Analyze (with model testing) all alignments in the
   ~/software/ParGenes/examples/data/small/fasta_files/
   folder using the default ParGenes settings

2*. Run model testing and tree inference for the prot21.fa
   alignment using 1 parsimony + 5 random starting trees.
Exercise 10: Answers

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

You will see warnings that some MSAs failed the check → That's fine!

[Warning] 2/9 commands failed
   Average number of taxa: 9
   Max number of taxa: 22
   Average number of sites: 1711
   Max number of sites: 6489
   Recommended MAXIMUM number of cores: 1
[Warning] Found 2 invalid MSAs (see parout/invalid_msas.txt)
[0:00:00] end of parsing mpi-scheduler run
[0:00:00] end of analysing parsing results

[Warning] Total number of jobs that failed: 3
[Warning] For a detailed list, see parout/failed_commands.txt
[0:00:54] END OF THE RUN OF pargenes.py
Exercise 10: Answers

$ python ~/software/ParGenes/pargenes/pargenes.py
   -a ~/software/ParGenes/examples/data/small/fasta_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
Outline

• RAxML-NG Intro
• Lab1: Basics
• RAxML-NG Parallelization
• Lab2: Parallelization
• Conclusions
Software availability

- **RAXML-NG**
  - Code: https://github.com/amkozlov/raxml-ng
  - Web server: https://raxml-ng.vital-it.ch/
  - Paper under revision, BioRxiv preprint available: https://doi.org/10.1101/447110

- **ModelTest-NG**
  - Code: https://github.com/ddarriba/modeltest
  - Manuscript in preparation

- **ParGenes**
  - Code: https://github.com/BenoitMorel/ParGenes
  - Published in *Bioinformatics (Morel et al. 2018)* https://doi.org/10.1093/bioinformatics/bty839
Which tool to use as of Jan’2019?

- **RAxML**
  - Features not yet supported by RAxML-NG, e.g. GTRCAT model, rapid bootstrapping ...

- **ExaML**
  - Concatenated supermatrices with GTRCAT

- **ParGenes**
  - Lots of gene trees, coalescent methods

- **RAxML-NG**
  - All other cases :)
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
Děkuji

Questions?
References


• 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