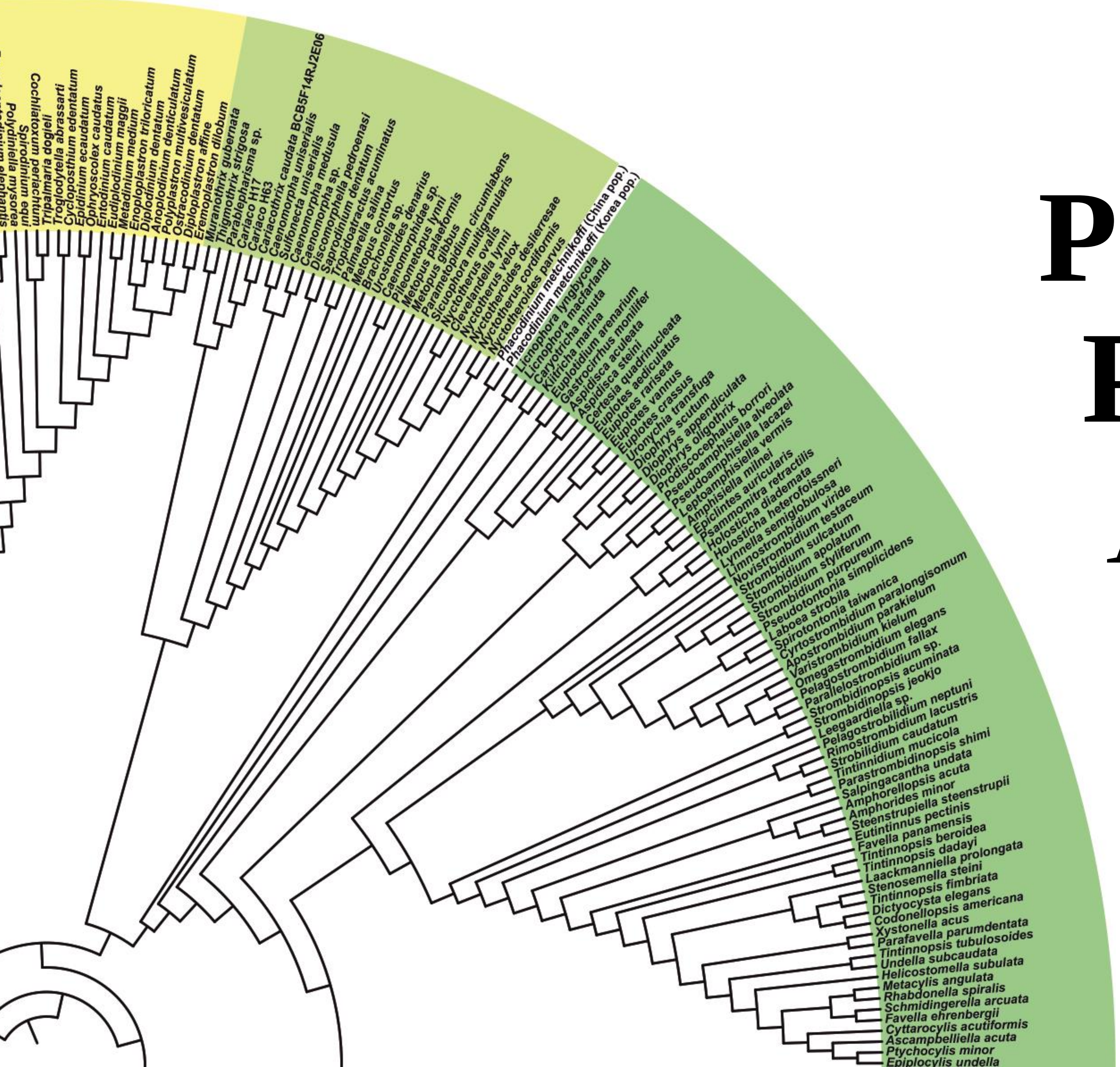
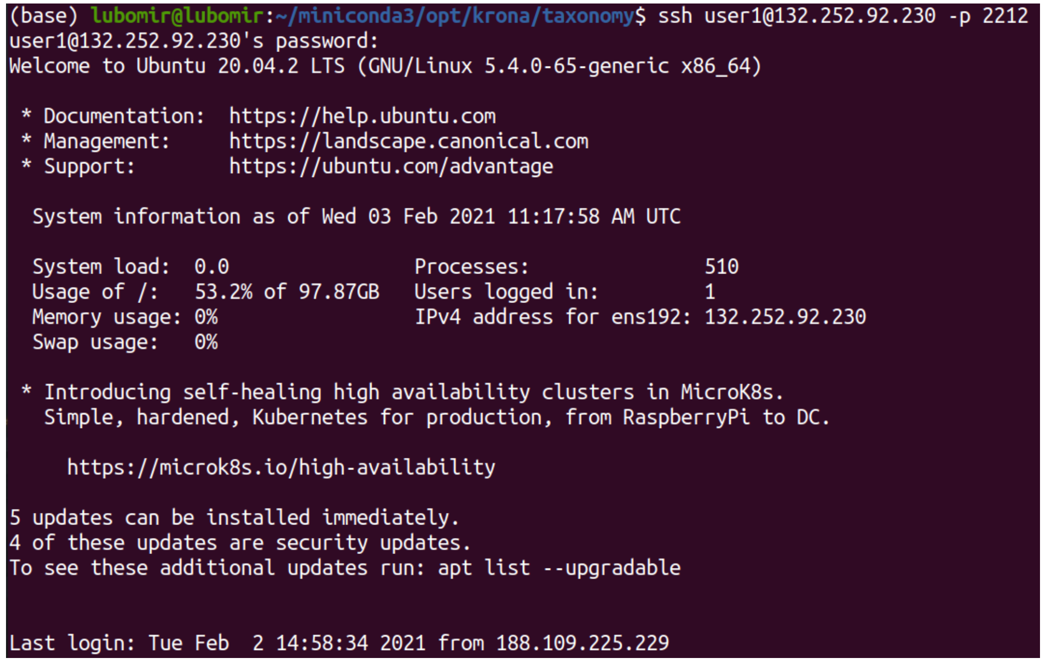


Phylogenetic Placement: A Practical Guide

Practical part

Unix environment

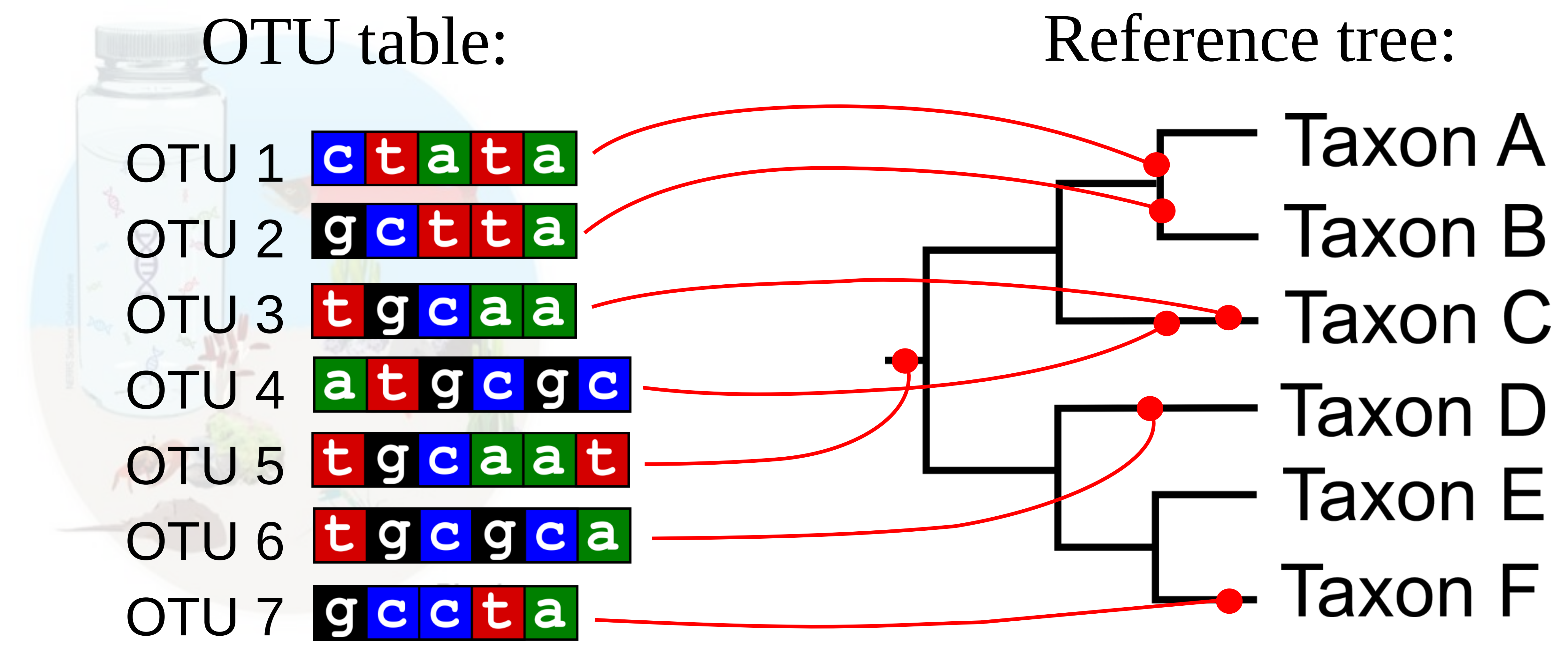


```
(base) lubomir@lubomir:~/miniconda3/opt/krona/taxonomy$ ssh user1@132.252.92.230 -p 2212
user1@132.252.92.230's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 03 Feb 2021 11:17:58 AM UTC

  System load:  0.0                Processes:               510
  Usage of /:   53.2% of 97.87GB   Users logged in:         1
  Memory usage: 0%                 IPv4 address for ens192: 132.252.92.230
  Swap usage:   0%

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

     https://microk8s.io/high-availability

5 updates can be installed immediately.
4 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Tue Feb  2 14:58:34 2021 from 188.109.225.229
```

Theoretical part

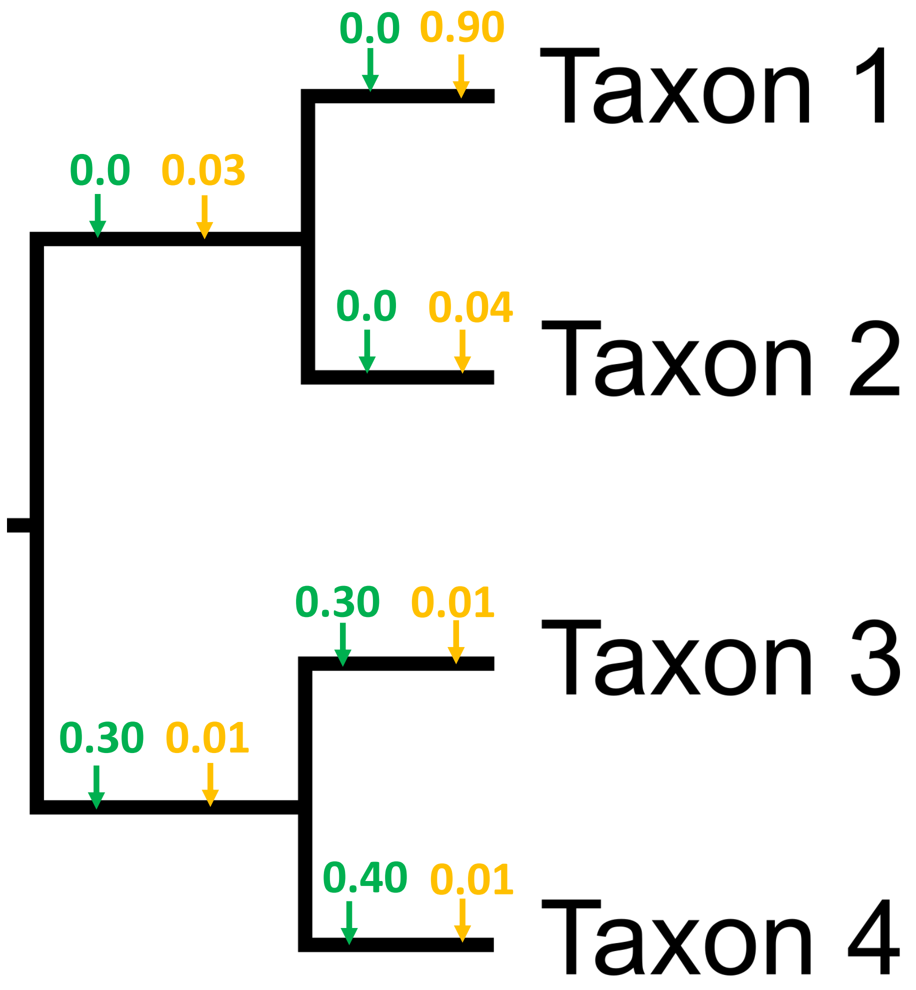
Presentation

Phylogenetic placement



Pic 1. is from Watts, A., & Miksis-Olds, J.L. (2018). The Ocean as a Living Sensor: Environmental DNA and Acoustics for Detecting Marine Life.

What the algorithm actually does



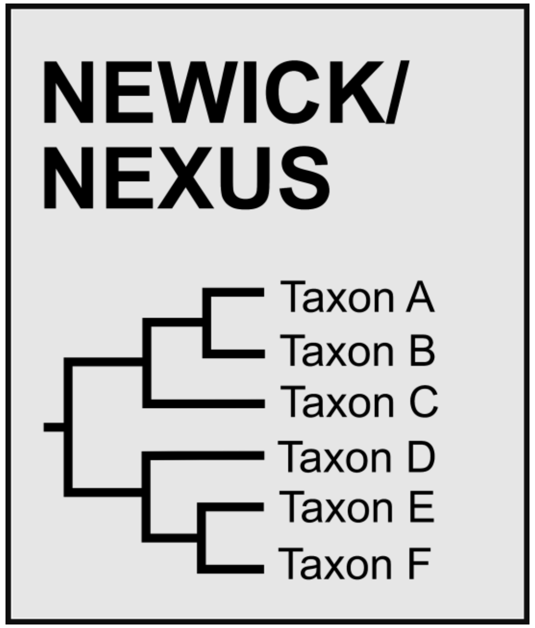
OTU_1 TGAAACAGTAATCGTAGCC

OTU_2 TGAAATCCCAATATGAGCC

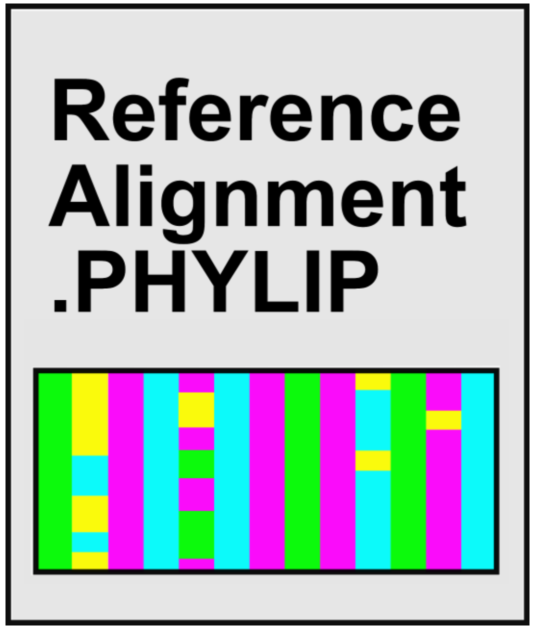
Likelihood Weight Ratio (LWR)
A confidence number of a given placement

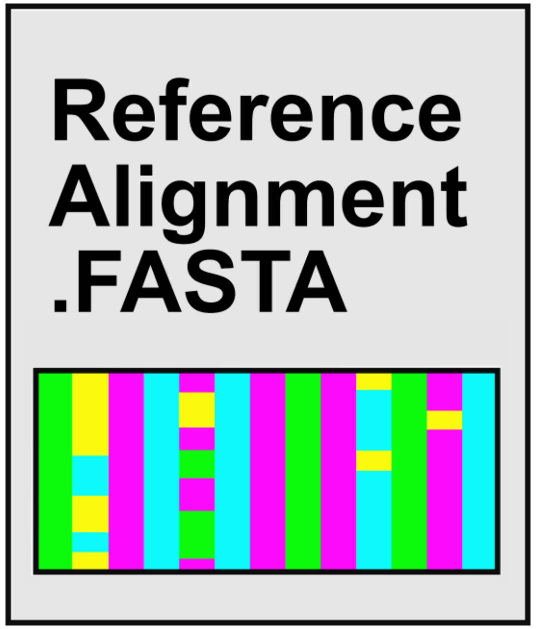
Input Data

1. Reference Tree

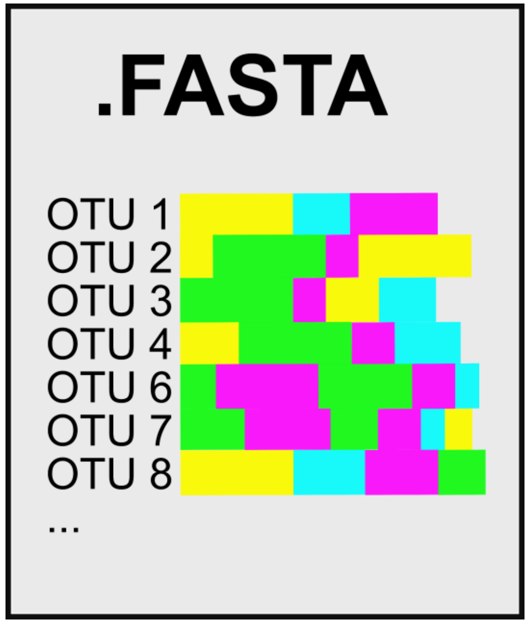


2. Reference Alignment





3. Query sequences



Pipeline

Align the query sequences based on the reference alignment

PAPARA

Compute parameters of the substitution model

RAxML-NG

Phylogenetic placement

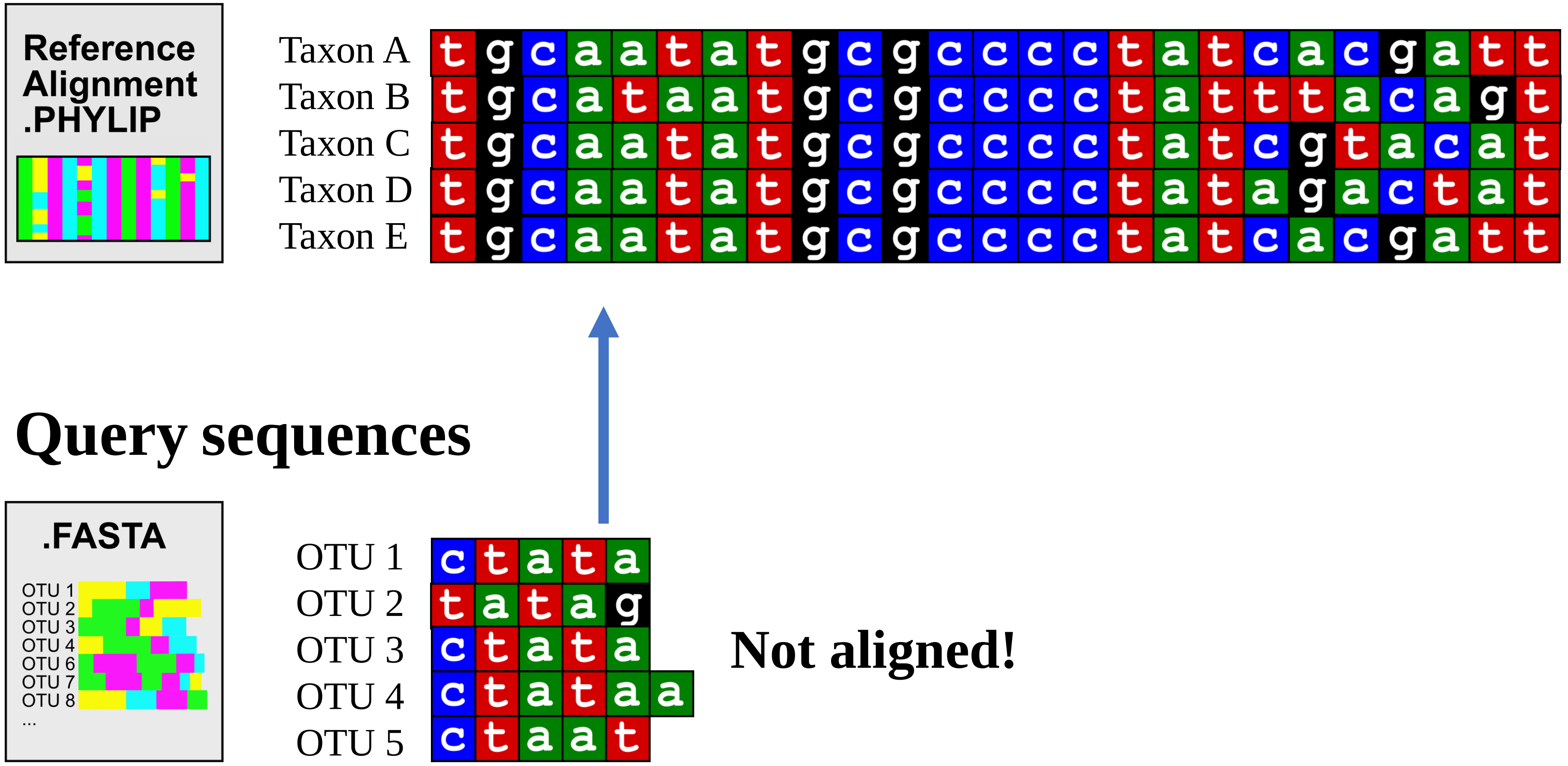
EPA-NG

Downstream analyses

GAPPA

Aligning step

Reference Alignment



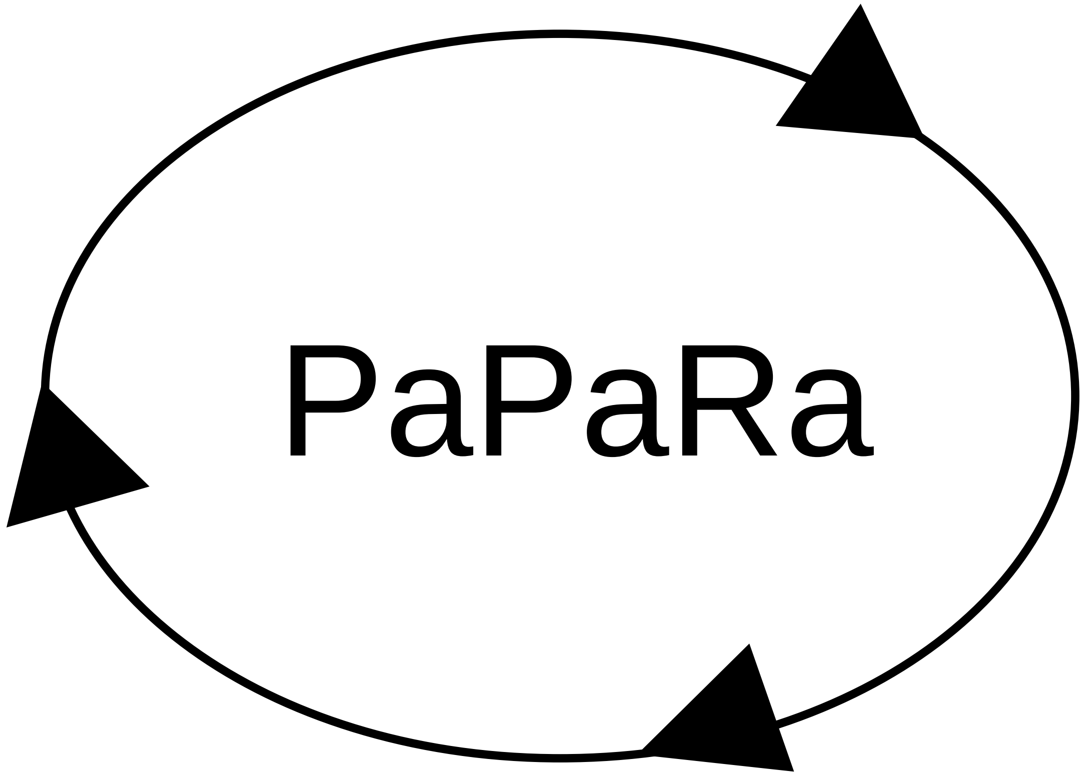
```
Taxon A  t g c a a t a t g c g c c c c t a t c a c g a t t
Taxon B  t g c a t a a t g c g c c c c t a t t t a c a g t
Taxon C  t g c a a t a t g c g c c c c t a t c g t a c a t
Taxon D  t g c a a t a t g c g c c c c t a t a g a c t a t
Taxon E  t g c a a t a t g c g c c c c t a t c a c g a t t
```

Already aligned!

Query sequences

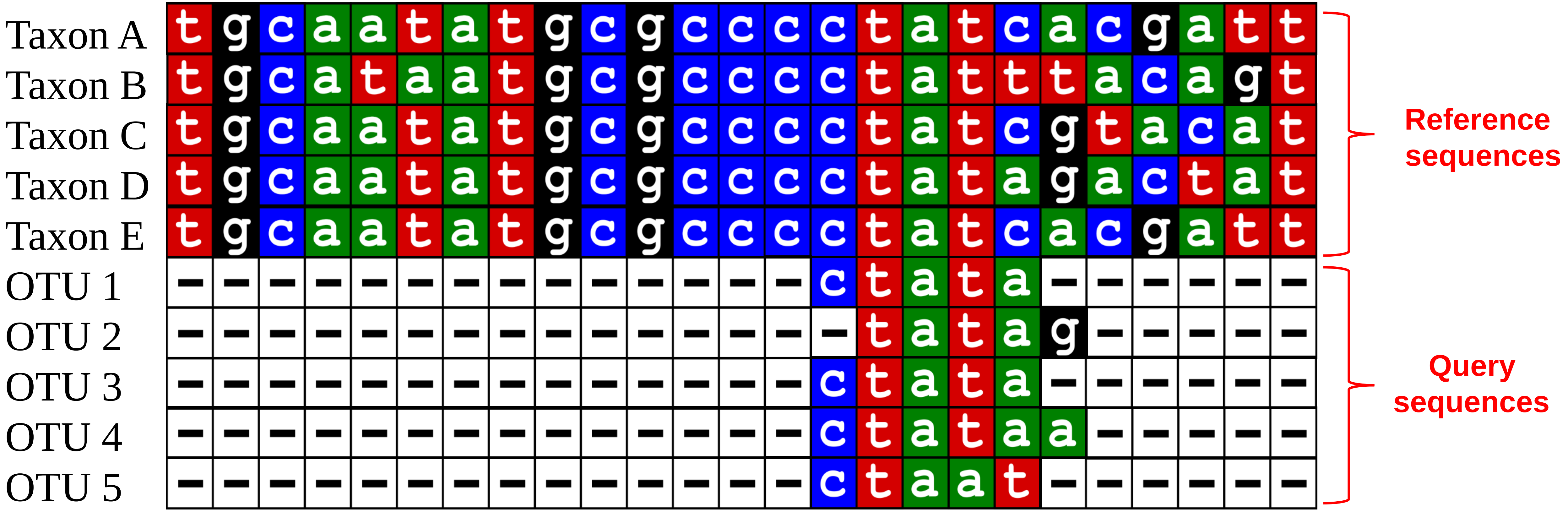
```
OTU 1  c t a t a
OTU 2  t a t a g
OTU 3  c t a t a
OTU 4  c t a t a a
OTU 5  c t a a t
```

Not aligned!



https://github.com/sim82/papara_nt

Papara Alignment


Reference sequences
Query sequences

Taxon A	t	g	c	a	a	t	a	t	g	c	g	c	c	c	c	t	a	t	c	a	c	g	a	t	t
Taxon B	t	g	c	a	t	a	a	t	g	c	g	c	c	c	c	t	a	t	t	t	a	c	a	g	t
Taxon C	t	g	c	a	a	t	a	t	g	c	g	c	c	c	c	t	a	t	c	g	t	a	c	a	t
Taxon D	t	g	c	a	a	t	a	t	g	c	g	c	c	c	c	t	a	t	a	g	a	c	t	a	t
Taxon E	t	g	c	a	a	t	a	t	g	c	g	c	c	c	c	t	a	t	c	a	c	g	a	t	t
OTU 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	c	t	a	t	a	-	-	-	-	-	-
OTU 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	t	a	t	a	g	-	-	-	-	-
OTU 3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	c	t	a	t	a	-	-	-	-	-	-
OTU 4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	c	t	a	t	a	a	-	-	-	-	-
OTU 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	c	t	a	a	t	-	-	-	-	-	-

Workflow: aligning

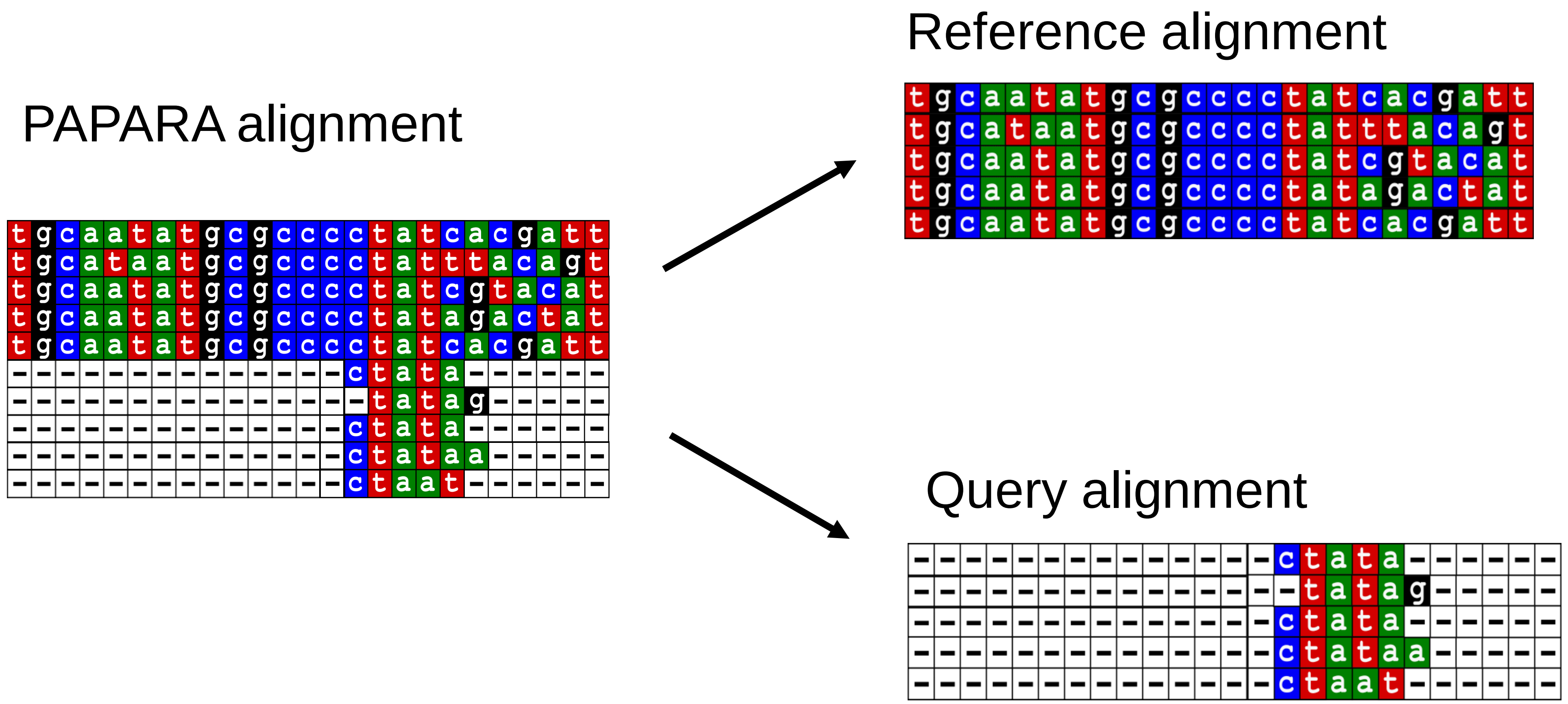
COMMAND:

```
papara -t reference_tree.newick -s reference_alignment.phylip -q
query_sequences.fasta -r
```

OUTPUT:

```
papara_alignment.default
papara_log.default
papara_quality.default
```

Papara Alignment needs to be split



PAPARA alignment

```
tgcaatatgcgcccctatcacgatt
tgcataatgcgcccctatttacagt
tgcaatatgcgcccctatcgtacat
tgcaatatgcgcccctatagactat
tgcaatatgcgcccctatcacgatt
--------------ctata------
---------------tatag-----
--------------ctata------
--------------ctataa-----
--------------ctaat------
```

Reference alignment

```
tgcaatatgcgcccctatcacgatt
tgcataatgcgcccctatttacagt
tgcaatatgcgcccctatcgtacat
tgcaatatgcgcccctatagactat
tgcaatatgcgcccctatcacgatt
```

Query alignment

```
--------------ctata------
---------------tatag-----
--------------ctata------
--------------ctataa-----
--------------ctaat------
```

Workflow: splitting alignment

COMMAND:

```
epa-ng --split reference_alignment.fasta papara_alignment.default
```

OUTPUT:

```
query.fasta
reference.fasta
```

Pipeline

Align the query sequences based on the reference alignment ✓

PAPARA

Compute parameters of the substitution model

RAxML-NG

Phylogenetic placement

EPA-NG

Downstream analyses

GAPPA

Substitution model and its parameters

Using in phylogenetic inferences

Has a meaning:

- Corrected hidden changes in DNA sequences

Various models

Each model has parameters that could be calculated

Example:

$$\underbrace{\text{GTR}}_{\text{Model}}\underbrace{\{0.5/2.0/1.0/1.2/0.1/1.0\}}_{\text{Substitution matrix}} + \underbrace{\text{FU}\{0.25/0.23/0.30/0.22\}}_{\text{Base frequency parameters}} + \ldots$$

Workflow: model evaluation

COMMAND:

```
raxml-ng --evaluate --msa reference.fasta --tree
reference_tree.newick --model GTR+G
```

OUTPUT:

```
reference.fasta.raxml.bestModel
reference.fasta.raxml.bestTree
reference.fasta.raxml.log
reference.fasta.raxml.rba
reference.fasta.raxml.startTree
```

Pipeline

Align the query sequences based on the reference alignment ✓
PAPARA

Compute parameters of the substitution model ✓
RAxML-NG

Phylogenetic placement
EPA-NG

Downstream analyses
GAPPA

Workflow: phylogenetic placement

COMMAND:

```
epa-ng -t reference_tree.newick -s reference.fasta -q query.fasta
--model reference.fasta.raxml.bestModel
```

OUTPUT:

```
epa_result.jplace
```

Pipeline

Align the query sequences based on the reference alignment ✓

PAPARA

Compute parameters of the substitution model ✓

RAxML-NG

Phylogenetic placement ✓

EPA-NG

Downstream analyses

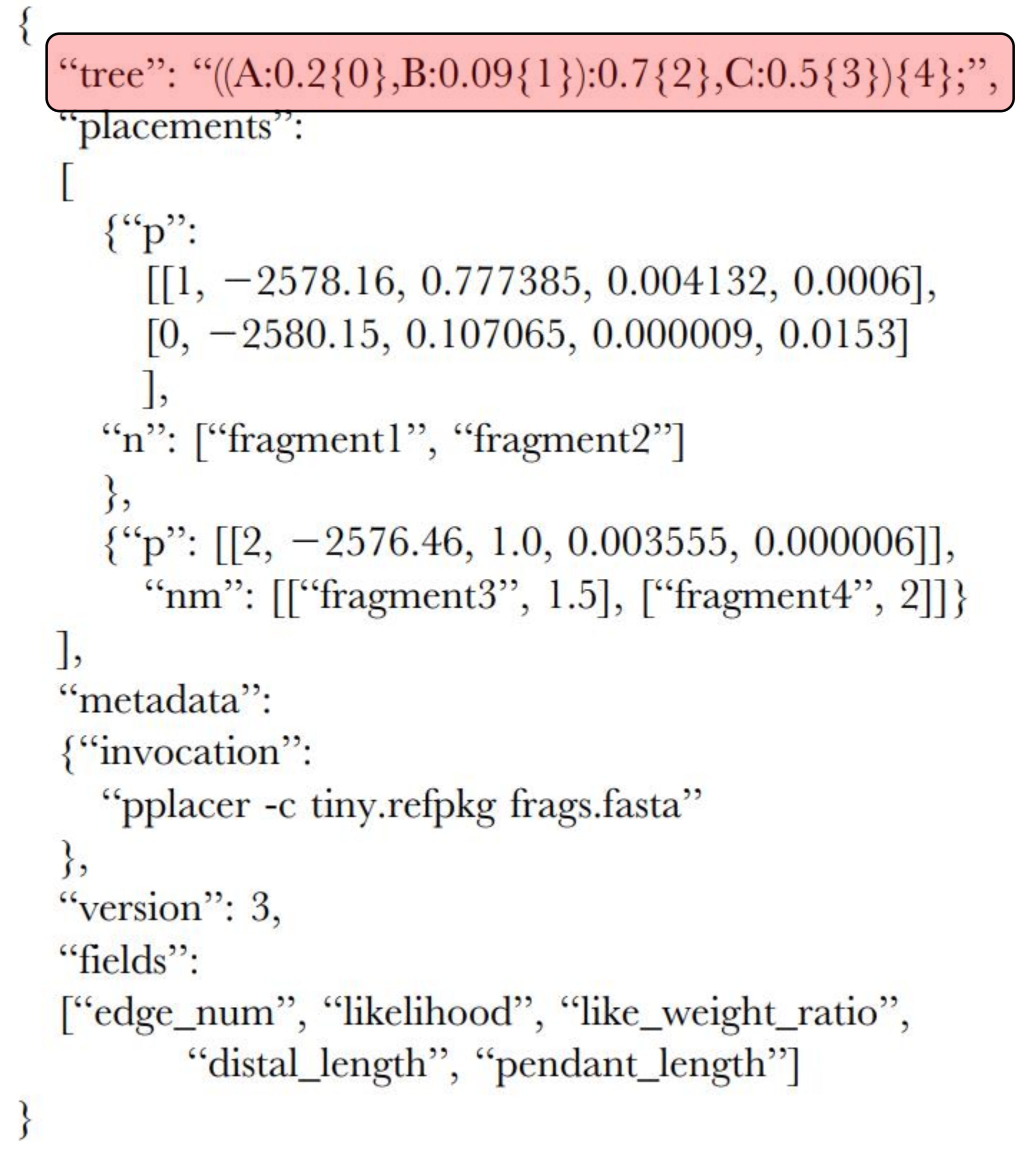
GAPPA

Example of a jPlace file

```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

https://github.com/Pbdas/epa-ng

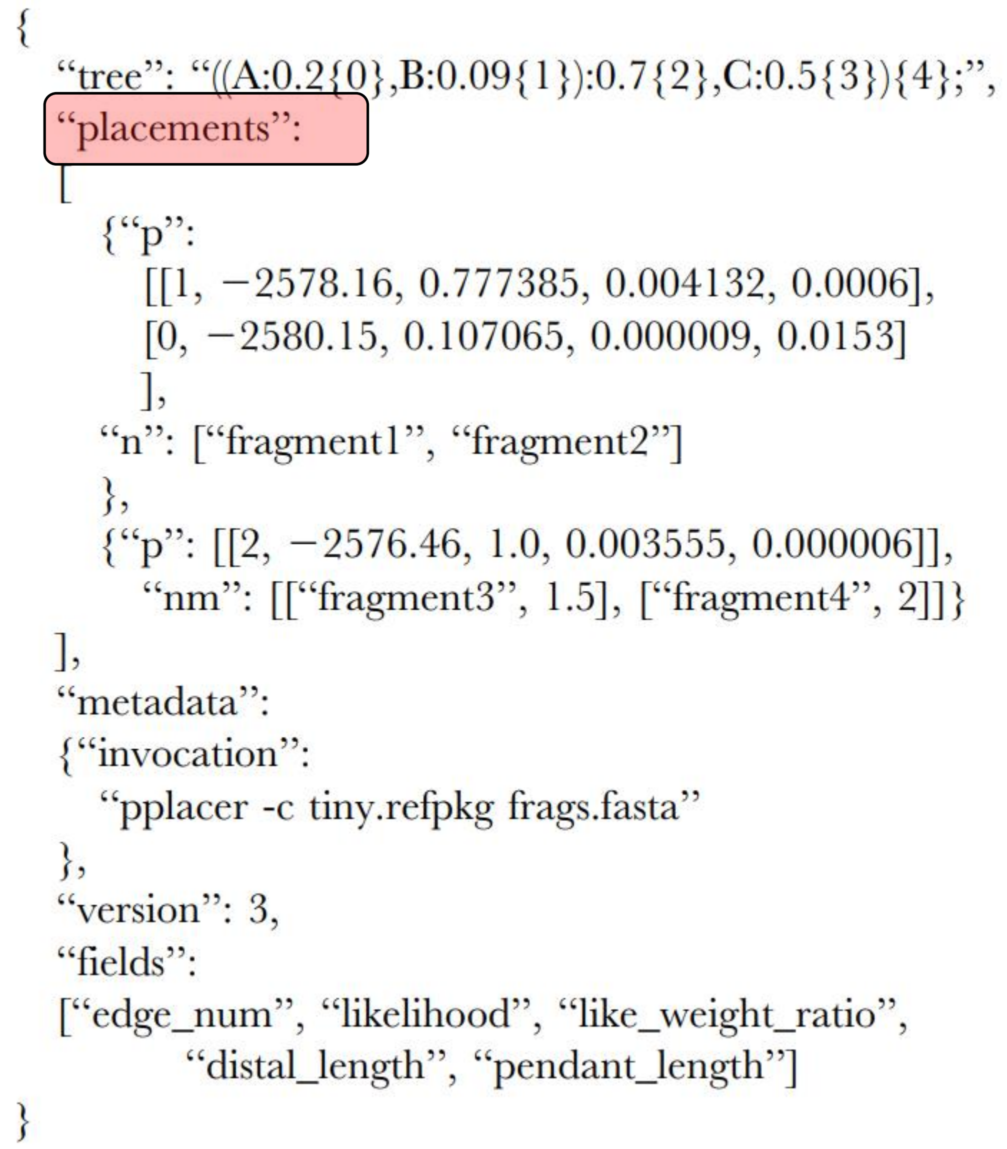
Example of a jPlace file



```
{
  “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
  “placements”:
  [
     {“p”:
        [[1, −2578.16, 0.777385, 0.004132, 0.0006],
        [0, −2580.15, 0.107065, 0.000009, 0.0153]
        ],
     “n”: [“fragment1”, “fragment2”]
     },
     {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
        “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
  ],
  “metadata”:
  {“invocation”:
     “pplacer -c tiny.refpkg frags.fasta”
  },
  “version”: 3,
  “fields”:
  [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ A tree in the Newick file format.

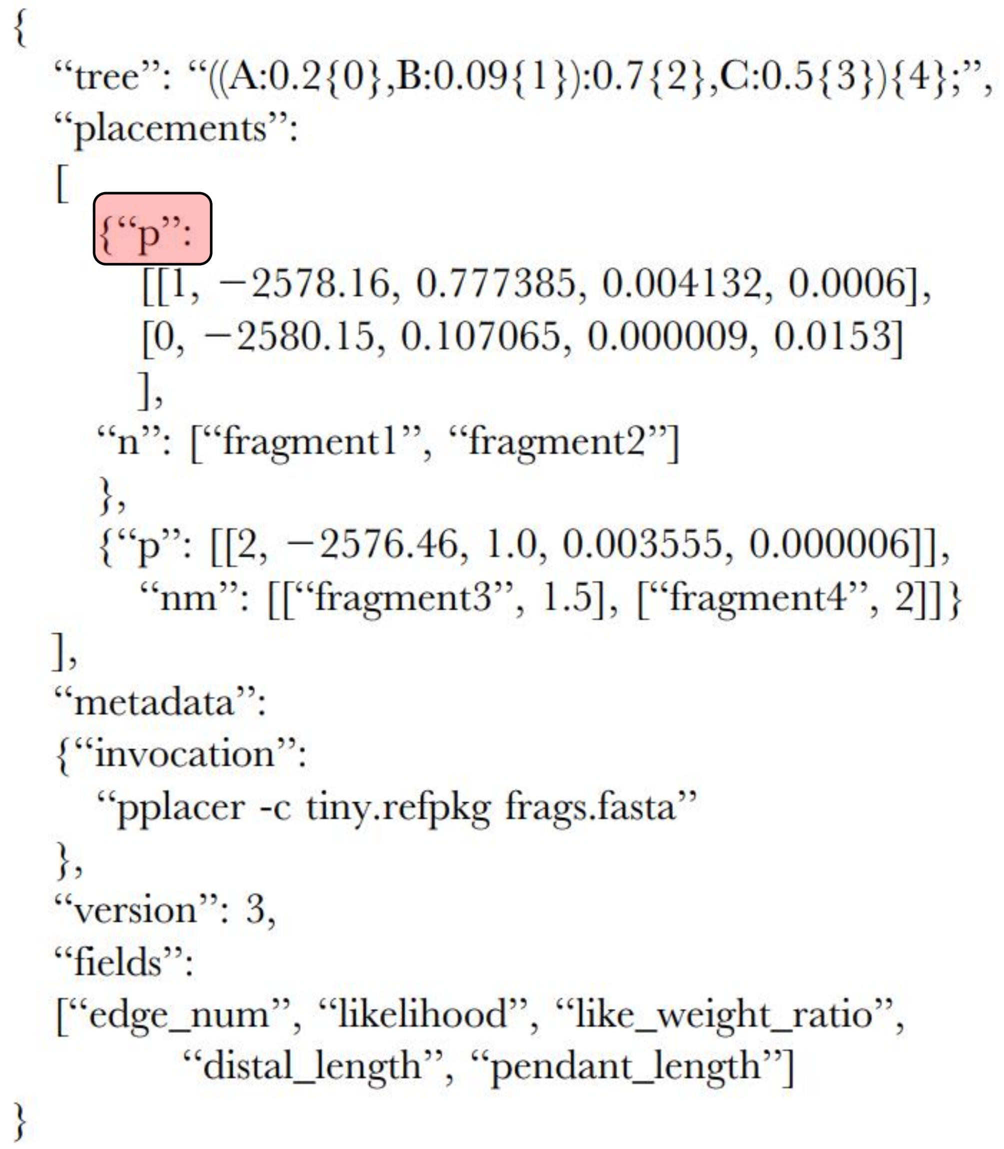
Example of a jPlace file



```
{
  “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
  “placements”:
  [
    {“p”:
      [[1, −2578.16, 0.777385, 0.004132, 0.0006],
      [0, −2580.15, 0.107065, 0.000009, 0.0153]
      ],
    “n”: [“fragment1”, “fragment2”]
    },
    {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
      “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
  ],
  “metadata”:
  {“invocation”:
    “pplacer -c tiny.refpkg frags.fasta”
  },
  “version”: 3,
  “fields”:
  [“edge_num”, “likelihood”, “like_weight_ratio”,
        “distal_length”, “pendant_length”]
}
```

List of placements

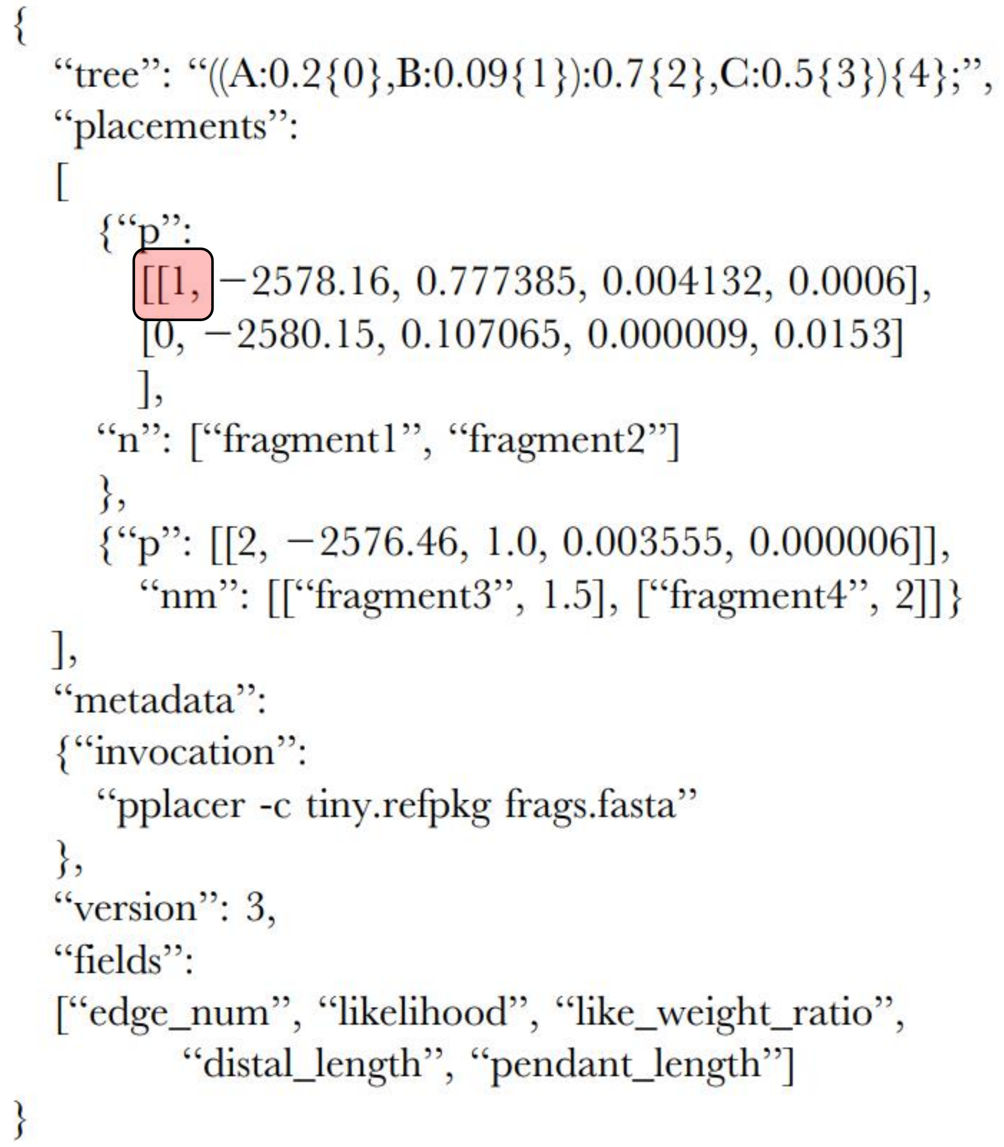
Example of a jPlace file



```
{
  “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
  “placements”:
  [
    {“p”:
      [[1, −2578.16, 0.777385, 0.004132, 0.0006],
      [0, −2580.15, 0.107065, 0.000009, 0.0153]
      ],
    “n”: [“fragment1”, “fragment2”]
    },
    {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
      “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
  ],
  “metadata”:
  {“invocation”:
    “pplacer -c tiny.refpkg frags.fasta”
  },
  “version”: 3,
  “fields”:
  [“edge_num”, “likelihood”, “like_weight_ratio”,
          “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ The list of placements shows possible placement locations along with their confidence scores and other information.

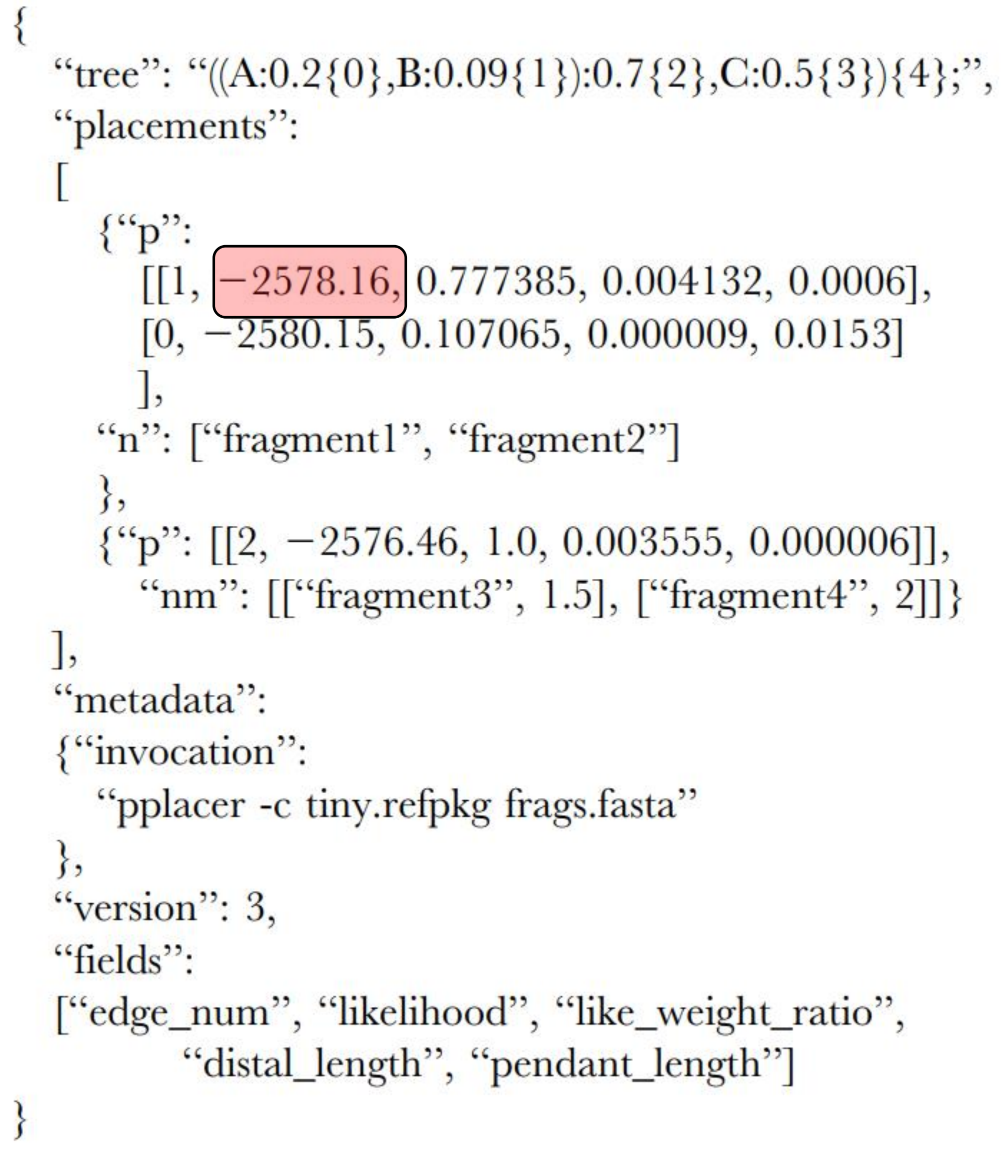
Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ **Edge Number** specifies the placement edge.

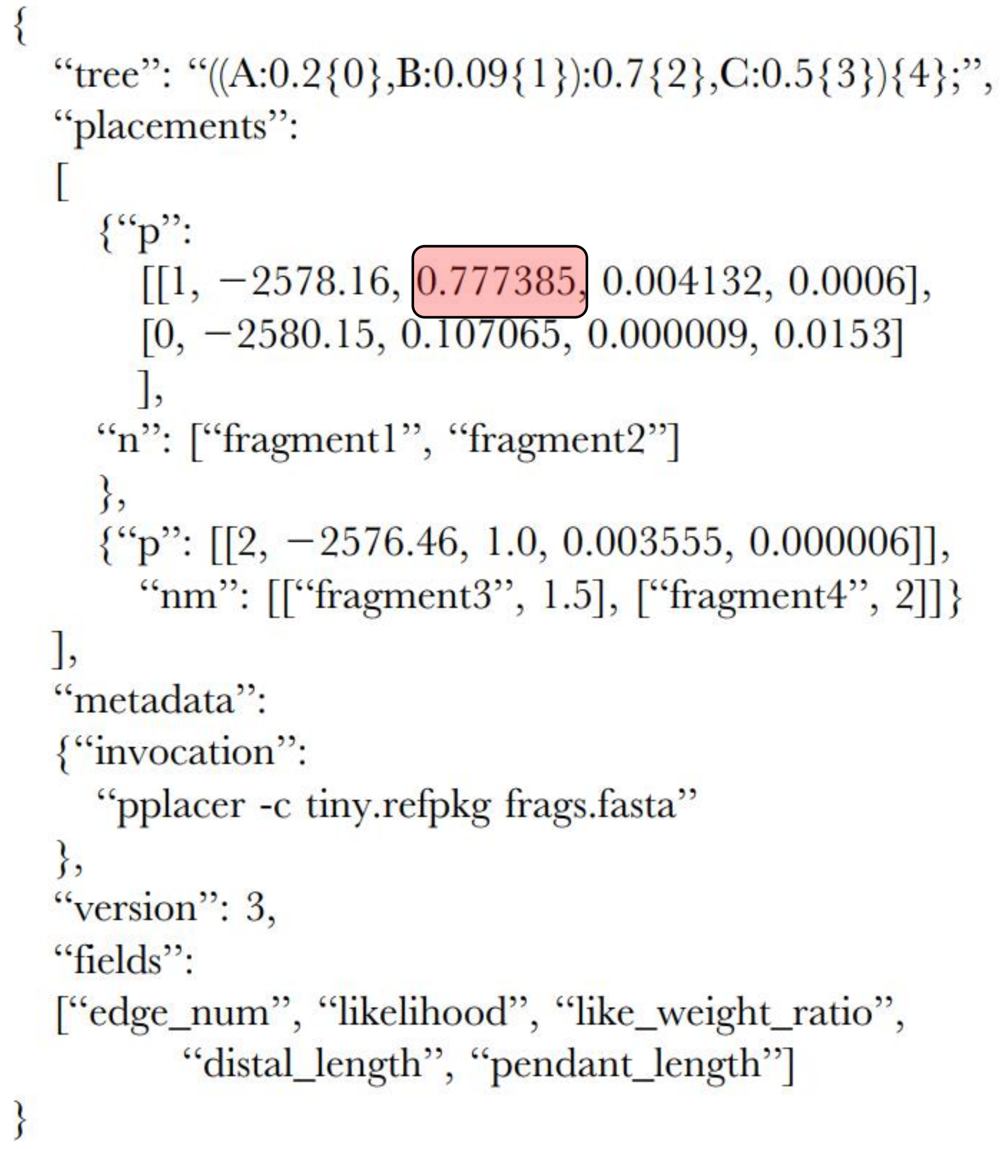
Example of a jPlace file



```
{
  "tree": "((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};",
  "placements":
  [
    {"p":
      [[1, −2578.16, 0.777385, 0.004132, 0.0006],
      [0, −2580.15, 0.107065, 0.000009, 0.0153]
      ],
    "n": ["fragment1", "fragment2"]
    },
    {"p": [[2, −2576.46, 1.0, 0.003555, 0.000006]],
      "nm": [["fragment3", 1.5], ["fragment4", 2]]}
  ],
  "metadata":
  {"invocation":
    "pplacer -c tiny.refpkg frags.fasta"
  },
  "version": 3,
  "fields":
  ["edge_num", "likelihood", "like_weight_ratio",
           "distal_length", "pendant_length"]
}
```

The **likelihood** of the tree with the placement attached.

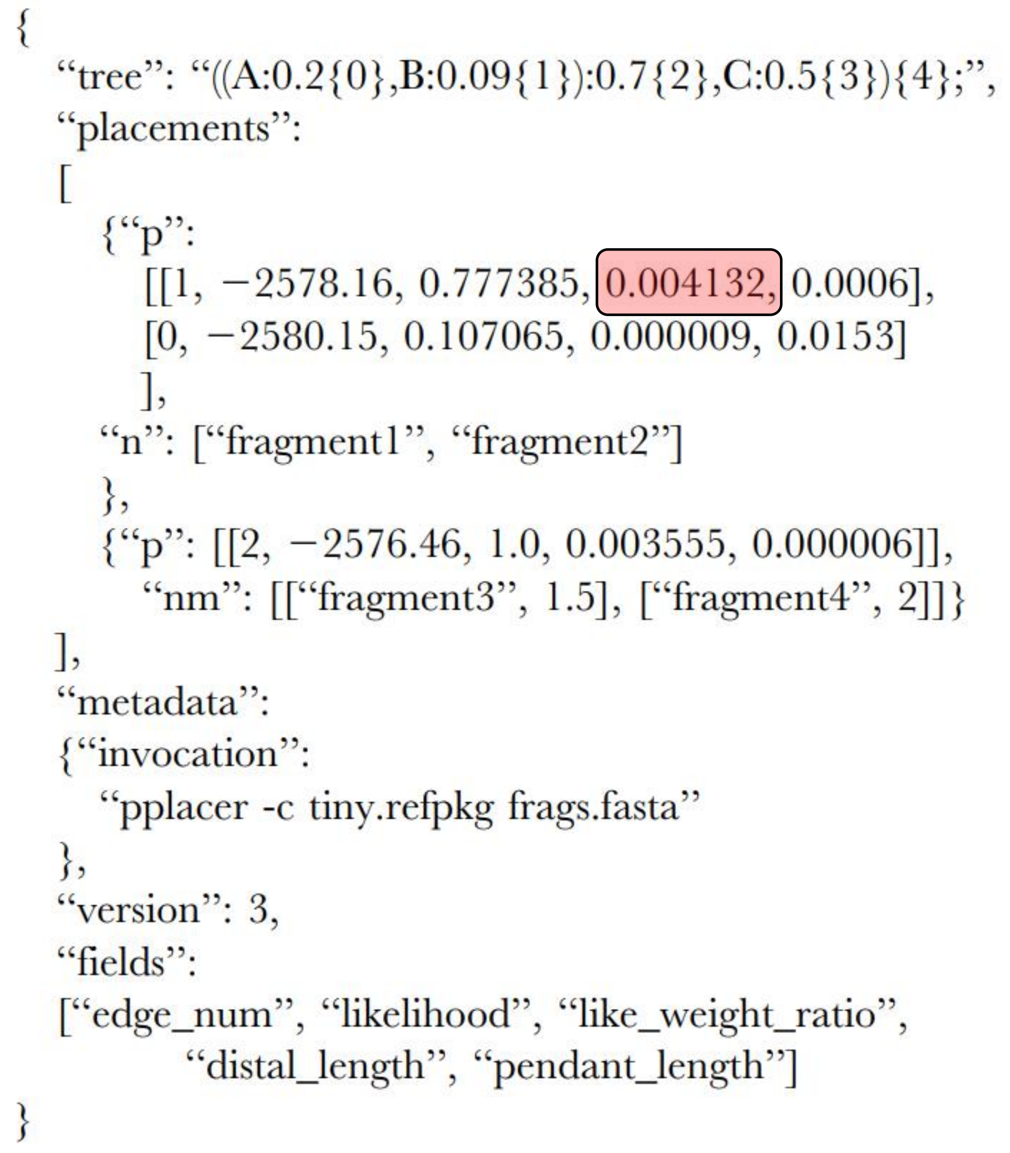
Example of a jPlace file



```
{
  “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
  “placements”:
  [
    {“p”:
      [[1, −2578.16, 0.777385, 0.004132, 0.0006],
      [0, −2580.15, 0.107065, 0.000009, 0.0153]
      ],
    “n”: [“fragment1”, “fragment2”]
    },
    {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
      “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
  ],
  “metadata”:
  {“invocation”:
    “pplacer -c tiny.refpkg frags.fasta”
  },
  “version”: 3,
  “fields”:
  [“edge_num”, “likelihood”, “like_weight_ratio”,
        “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ The **like weight ratio** is the ratio of that placement’s likelihood to that of the other alternate placements for that read.

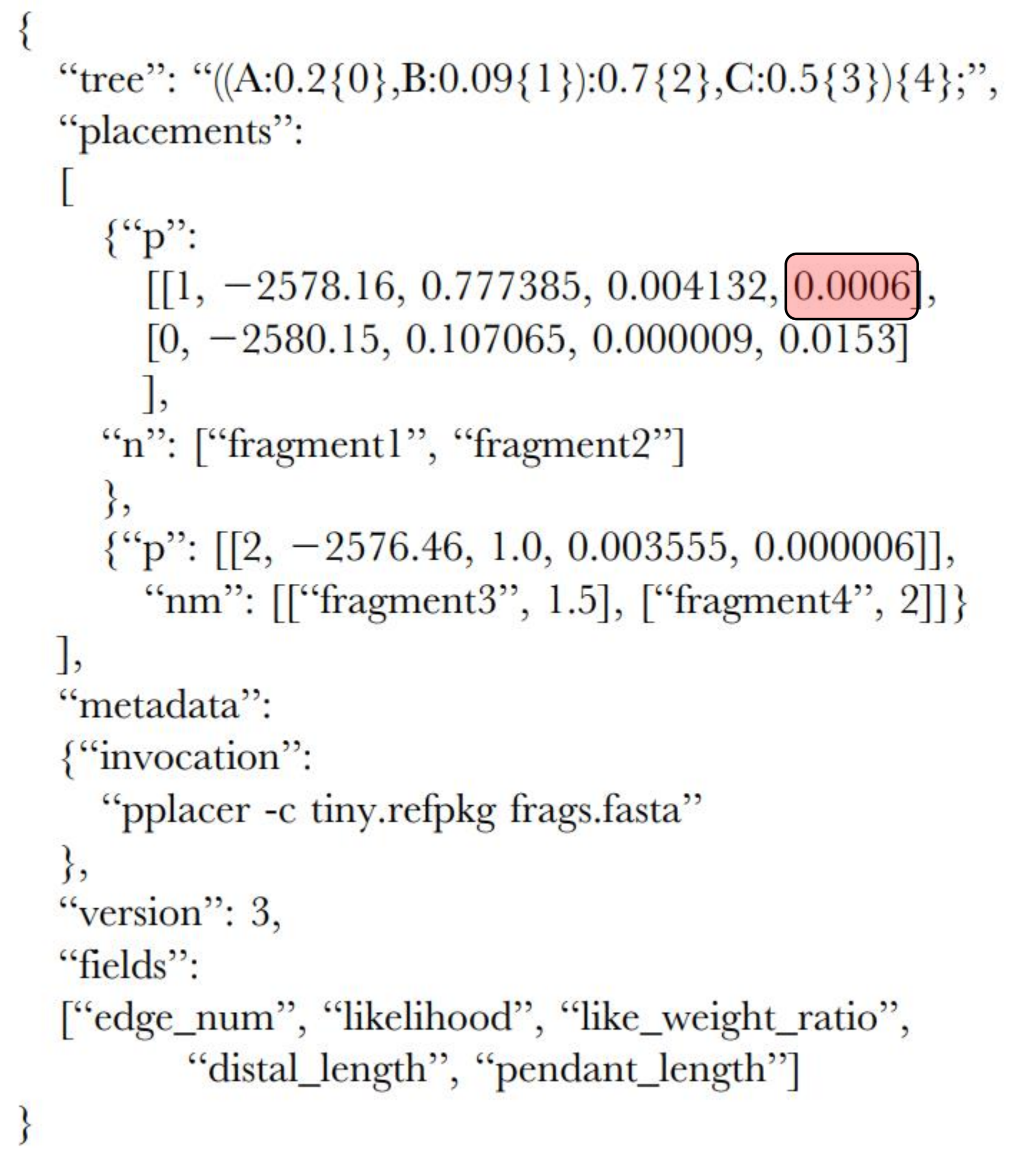
Example of a jPlace file



```
{
  “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
  “placements”:
  [
    {“p”:
      [[1, −2578.16, 0.777385, 0.004132, 0.0006],
      [0, −2580.15, 0.107065, 0.000009, 0.0153]
      ],
    “n”: [“fragment1”, “fragment2”]
    },
    {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
      “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
  ],
  “metadata”:
  {“invocation”:
    “pplacer -c tiny.refpkg frags.fasta”
  },
  “version”: 3,
  “fields”:
  [“edge_num”, “likelihood”, “like_weight_ratio”,
        “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ **Distal length** is the length from the distal (away from the root) side of the reference tree edge to the placement attachment location.

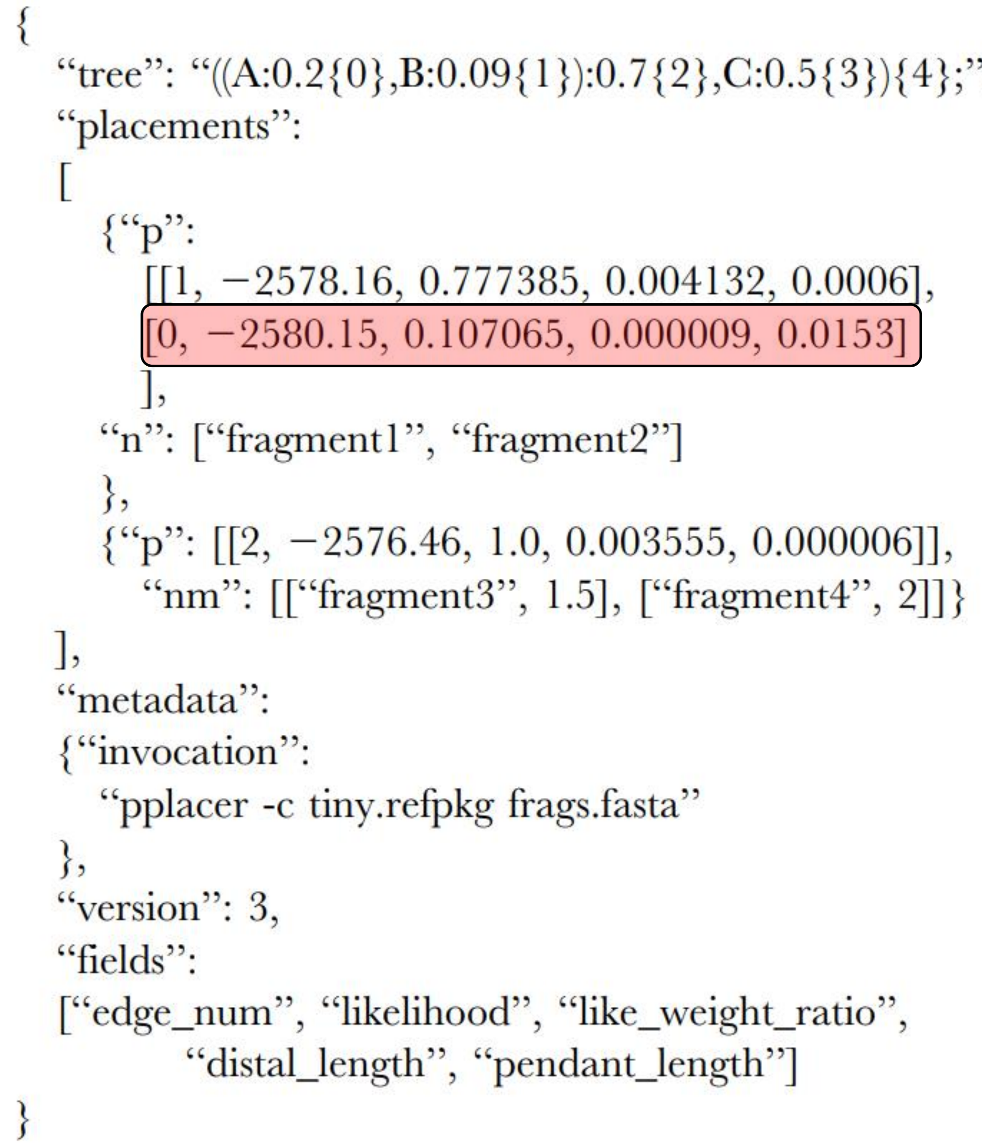
Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ **Pendant length** is the branch length for the placement edge

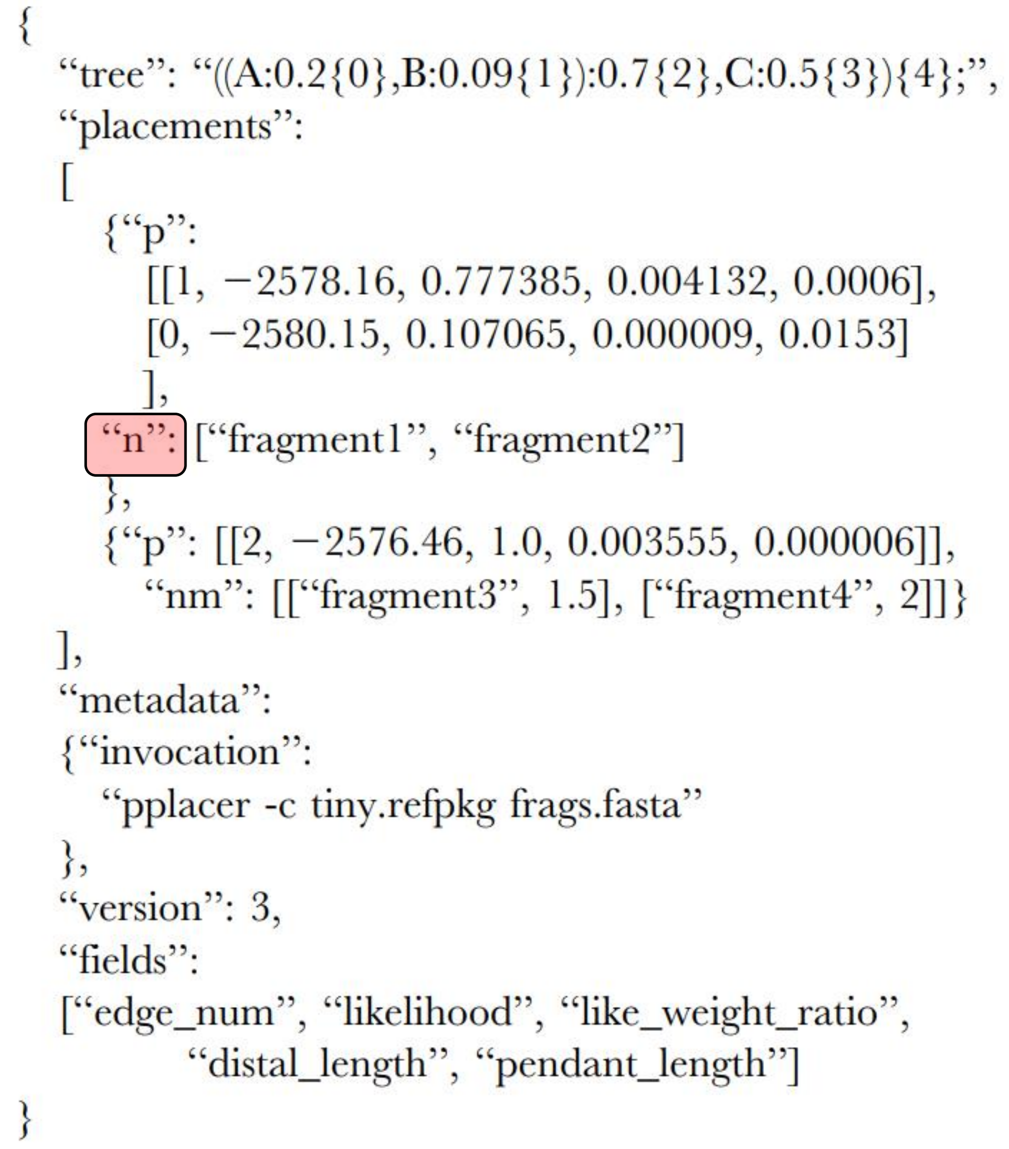
Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

$\longrightarrow$ Same goes for the next LWR placement of the same OTU.

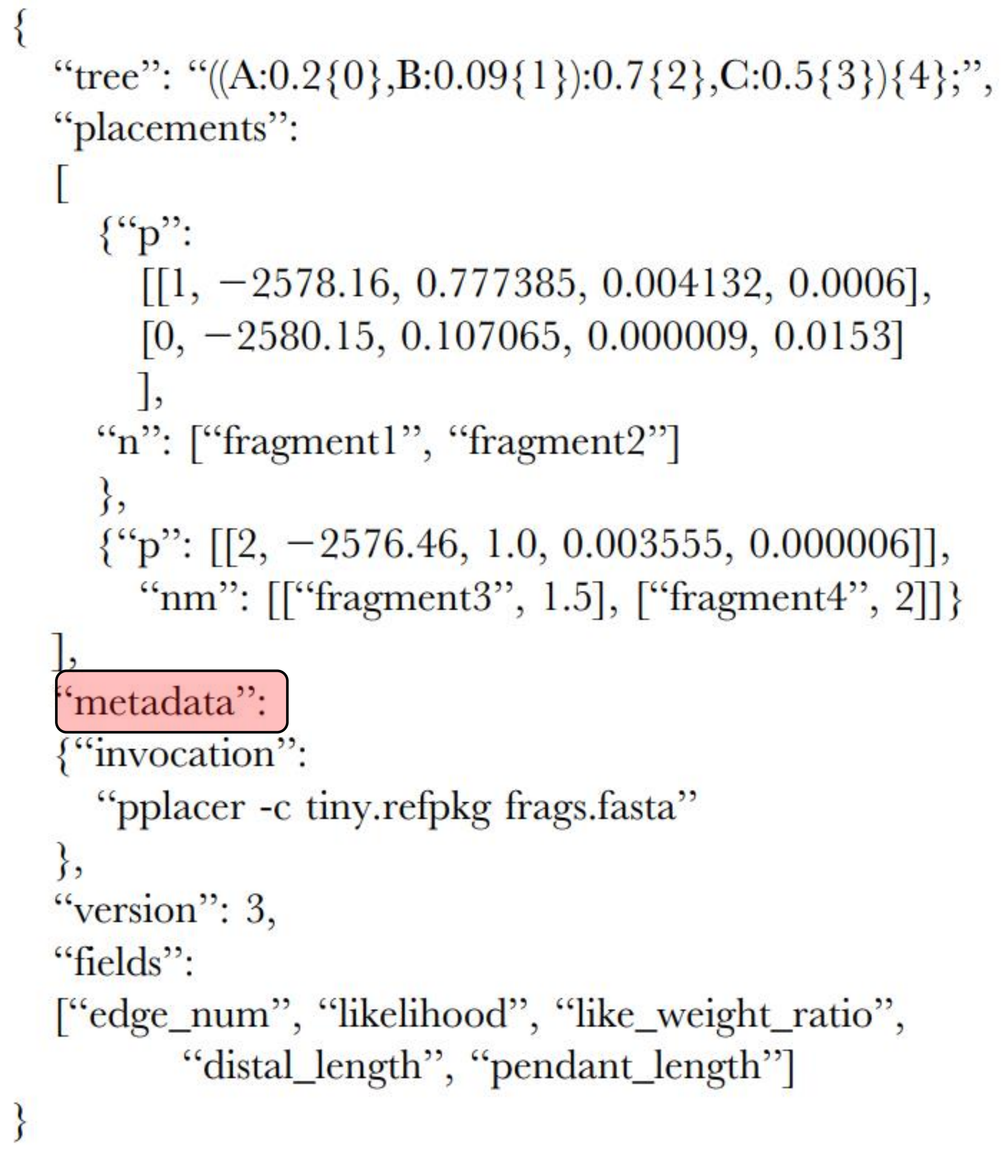
Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
         “distal_length”, “pendant_length”]
}
```

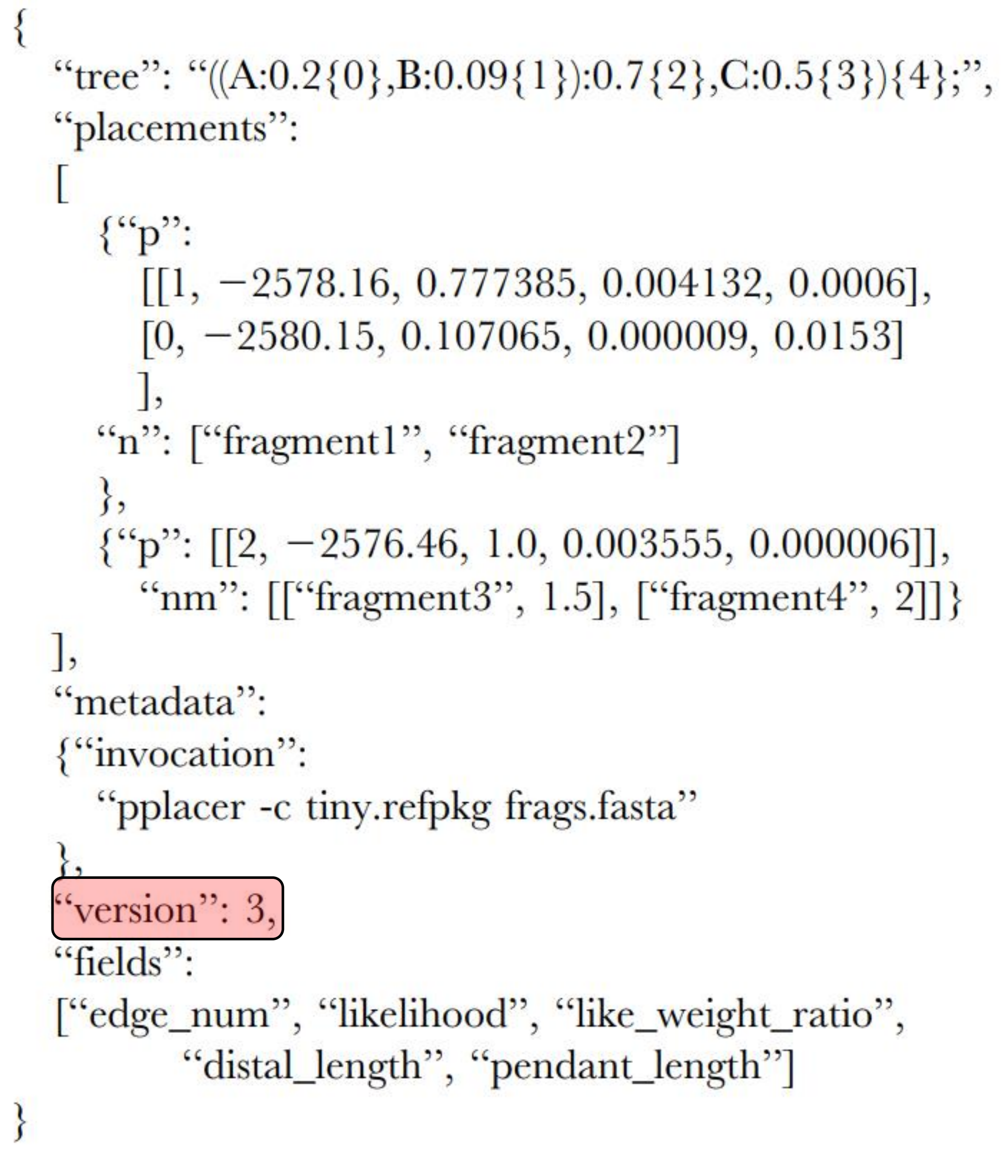
→ **OTU name** associated with placements.

Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
          “distal_length”, “pendant_length”]
}
```

Example of a jPlace file



```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
            “distal_length”, “pendant_length”]
}
```

Example of a jPlace file

```
{
   “tree”: “((A:0.2{0},B:0.09{1}):0.7{2},C:0.5{3}){4};”,
   “placements”:
   [
      {“p”:
         [[1, −2578.16, 0.777385, 0.004132, 0.0006],
         [0, −2580.15, 0.107065, 0.000009, 0.0153]
         ],
      “n”: [“fragment1”, “fragment2”]
      },
      {“p”: [[2, −2576.46, 1.0, 0.003555, 0.000006]],
         “nm”: [[“fragment3”, 1.5], [“fragment4”, 2]]}
   ],
   “metadata”:
   {“invocation”:
      “pplacer -c tiny.refpkg frags.fasta”
   },
   “version”: 3,
   “fields”:
   [“edge_num”, “likelihood”, “like_weight_ratio”,
         “distal_length”, “pendant_length”]
}
```

What we can do with the output file?

GAPPA toolkit: https://github.com/lczech/gappa

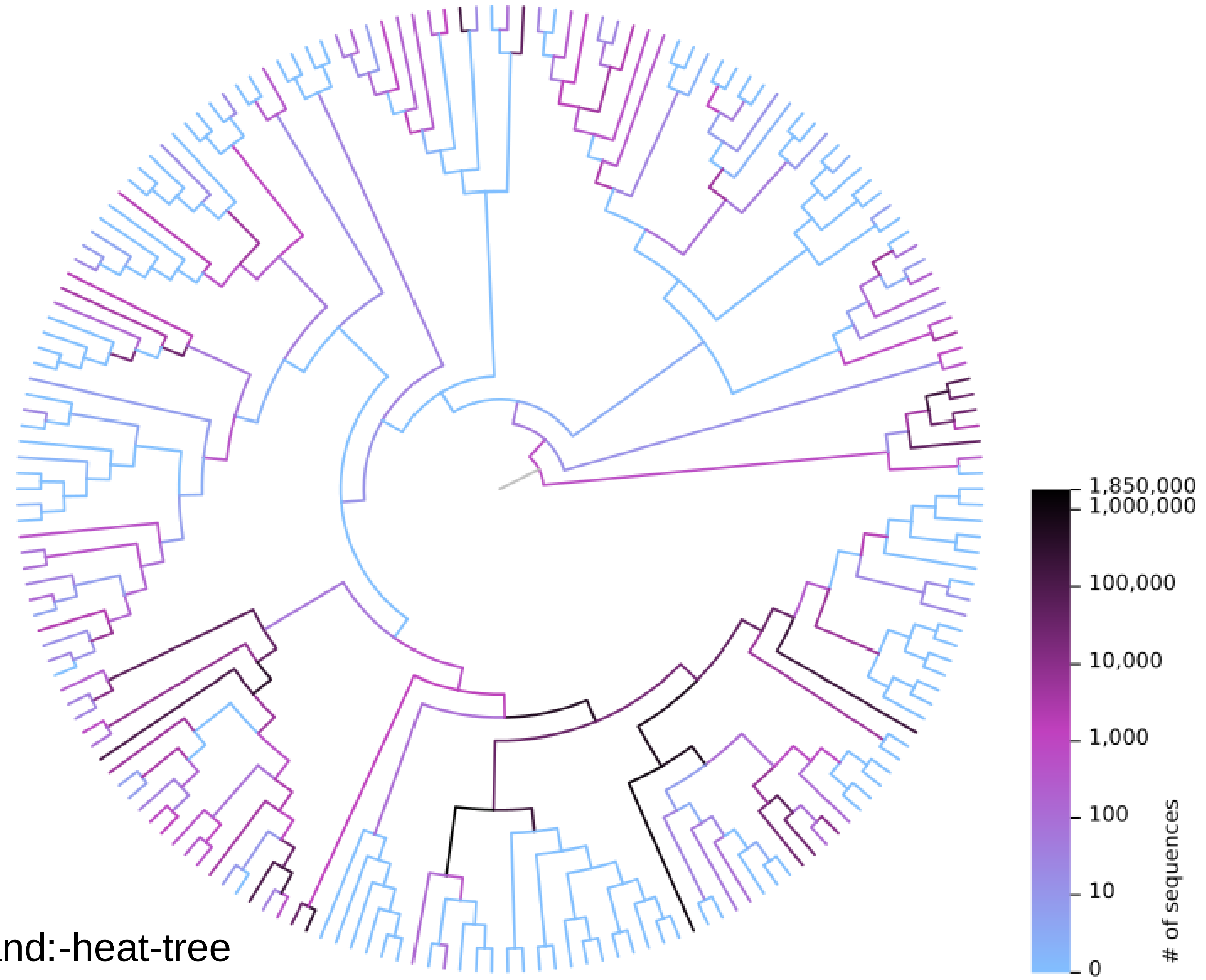
- Heat tree
- Taxonomic assignment
- Labelled tree
- Extraction to predefined groups
- Histograms

For more things to do, visit - https://github.com/lczech/gappa/wiki

Heat Tree

Commad *examine heat-tree* makes a tree with edges colored according to the placement mass of the samples.

Input is jplace and output is a tree in svg and nexus format.



https://github.com/lczech/gappa/wiki/Subcommand:-heat-tree

Workflow: heat tree

COMMAND:

```
gappa examine heat-tree --jplace-path epa_result.jplace --mass-
norm absolute --write-svg-tree --write-newick-tree --write-nexus-
tree
```

OUTPUT:

tree.svg

tree.nexus

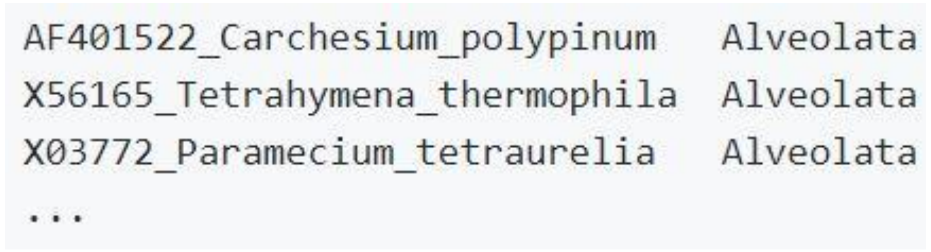
tree.newick

Taxonomic assignment

Taxonomically assign placed query sequences

We need two input files:

- jplace file from the phylogenetic placement
- taxon file containing a tab-separated list of reference taxon to taxonomic string assignments



```
AF401522_Carchesium_polypinum   Alveolata
X56165_Tetrahymena_thermophila  Alveolata
X03772_Paramecium_tetraurelia   Alveolata
...
```

More here - https://github.com/lczech/gappa/wiki/Subcommand:-assi

Workflow: taxonomical assignment

COMMAND:

```
gappa examine assign --jplace-path epa_result.jplace --taxon-file clades.txt --krona
```

output:

```
assign_krona.profile
assign_labelled_tree.newick
assign_profile.tsv
```

How to interpret the output

Final output is profile.tsv

The meaning of the column headers are:

- **LWR:** likelihood weight that was assigned to this exact taxonomic path
- **fract:** LWR divided by the global total likelihood weight
- **aLWR:** accumulated likelihood weights that were assigned either to this taxonomic path or any taxonomic path below this
- **afract:** aLWR divided by the global total likelihood weight
- **taxopath:** the taxonomic path

Advanced – Krona Output

You need to install Krona Tools https://github.com/marbl/Krona/tree/master/KronaTools

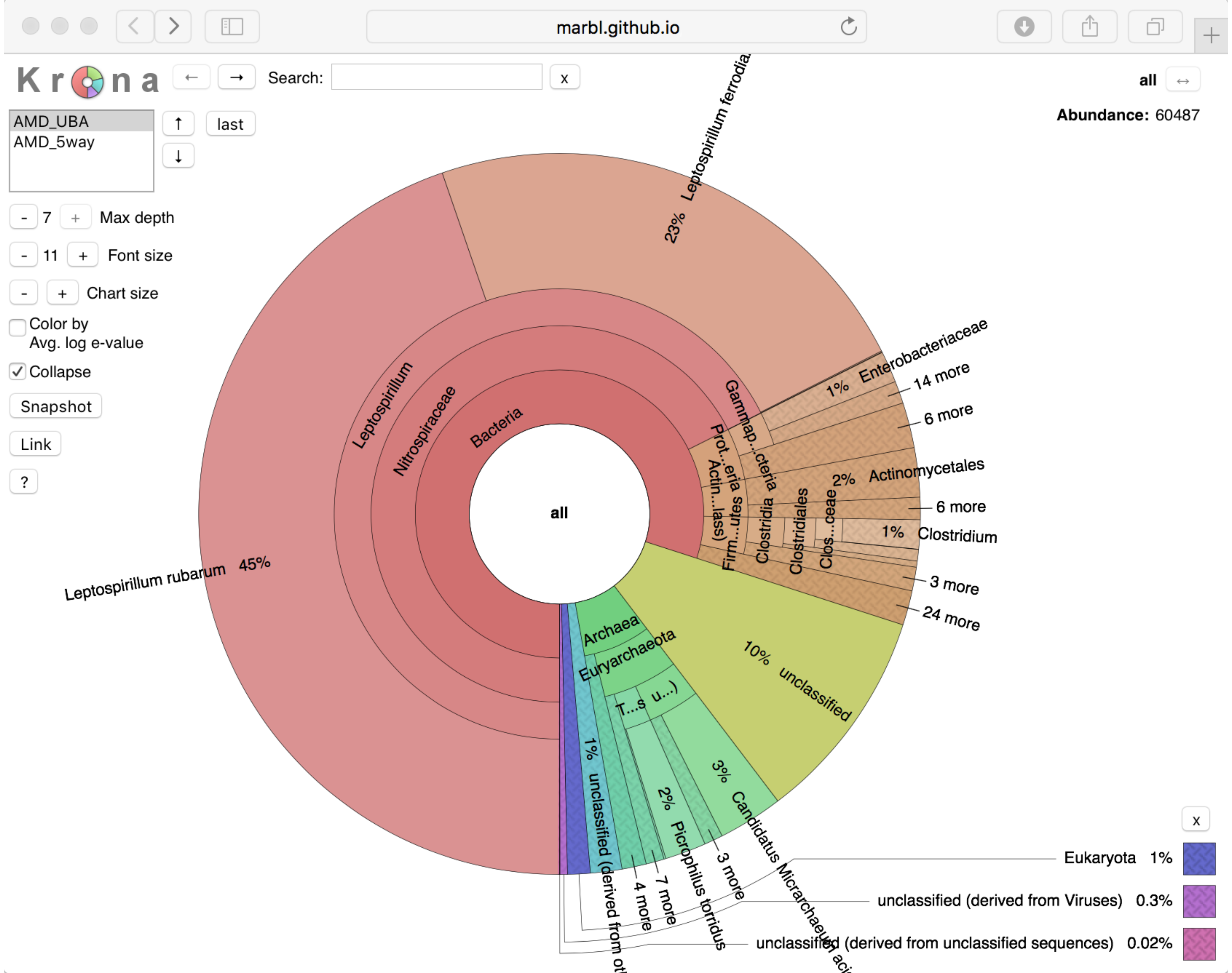
But if you have bioconda, you can install it simply by using: conda install -c bioconda krona

Then, you can create a taxonomic chart from the assign_krona.profile file using: ktImportText assign_krona.profile

More here: https://github.com/marbl/Krona/wiki/Importing-text-and-XML-data#text

Finally, you can open and edit the taxonomy chart in a web browser

Krona taxonomy chart



- 7 + Max depth

- 11 + Font size

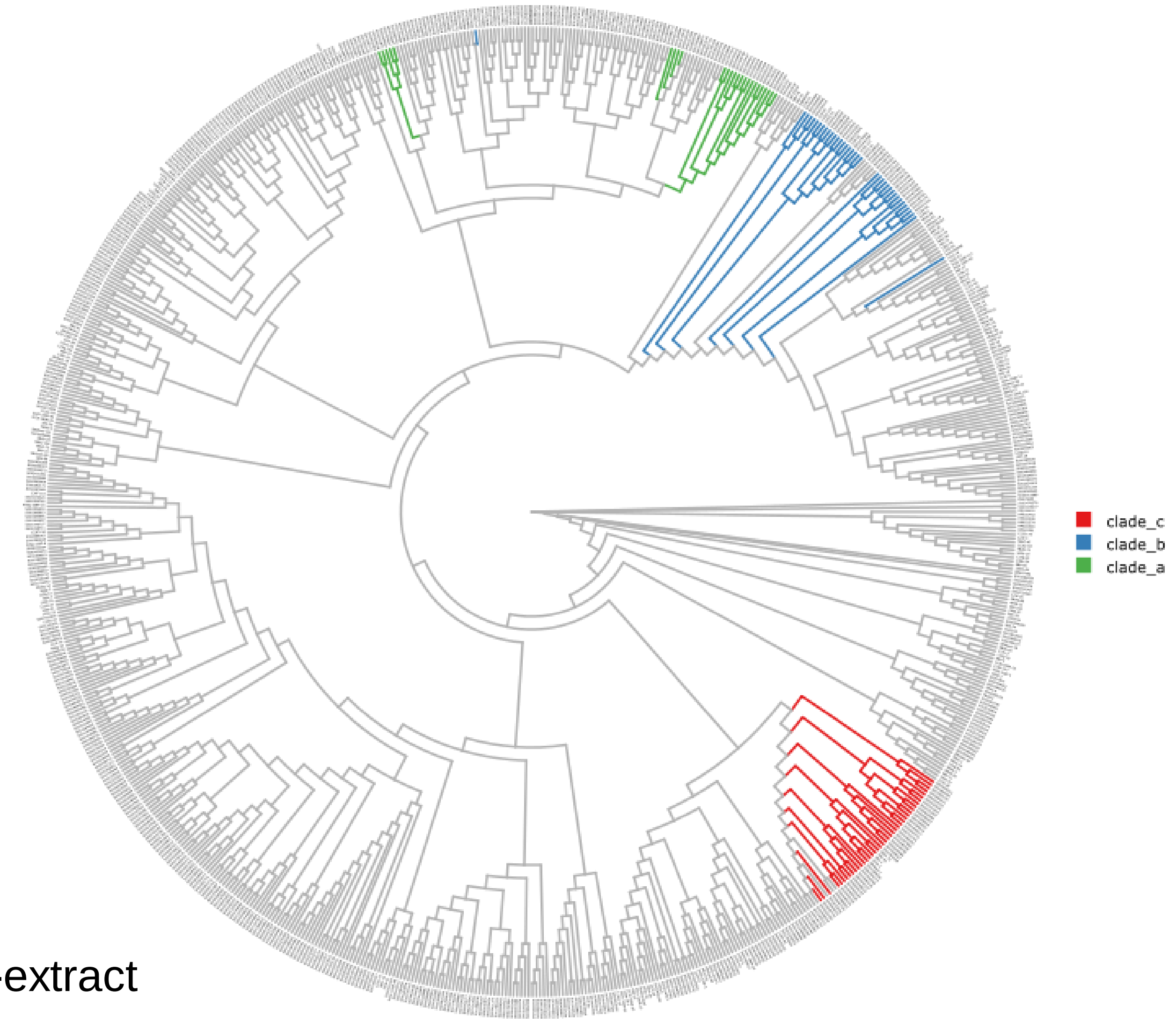
- + Chart size

[ ] Color by Avg. log e-value

[✓] Collapse

Extraction of OTUs in GAPPA

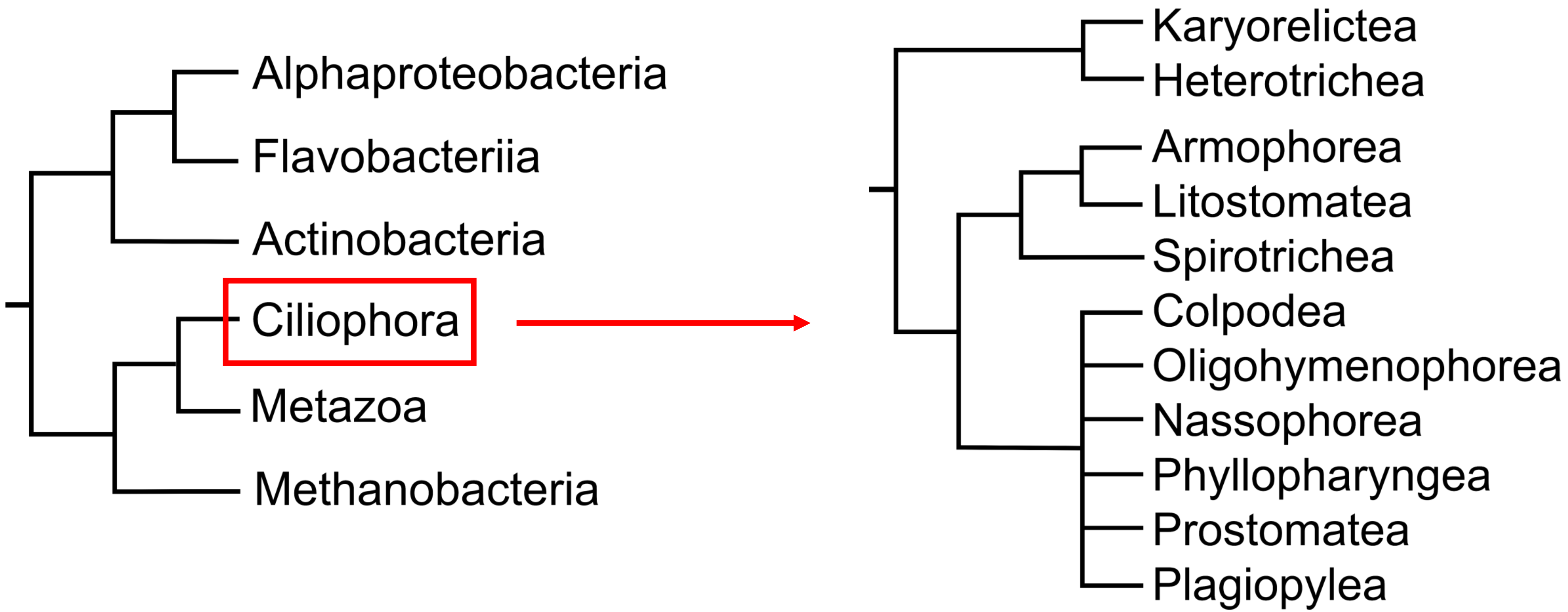
prepare extract command extracts placements from clades of the tree and write per-clade jplace files



https://github.com/lczech/gappa/wiki/Subcommand:-extract

Extraction of OTUs

Broad-taxa range **Taxon-specific**

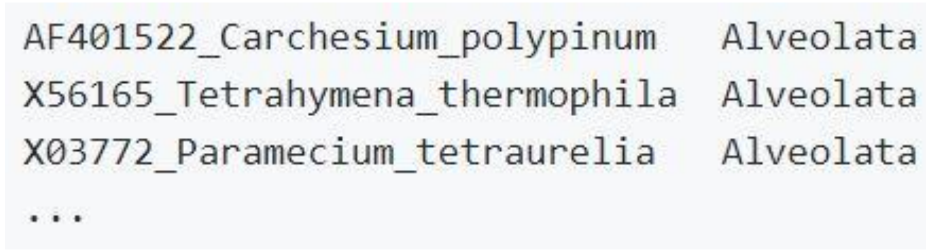


Input Files

We need three input files:

1. **jplace file** from the epa analysis

2. **clade list** – a text file containing a tab-separated list of taxa and clades upon witch will be make the OTU extraction



```
AF401522_Carchesium_polypinum   Alveolata
X56165_Tetrahymena_thermophila  Alveolata
X03772_Paramecium_tetraurelia   Alveolata
...
```

3. **fasta file** with all OTU sequences

Workflow: extraction

Command:

```
gappa prepare extract --jplace-path epa_result.jplace --clade-list-file clades.txt --fasta-path query.fasta --color-tree-file extract_tree --samples-out-dir samples --sequences-out-dir sequences
```

OUTPUT:

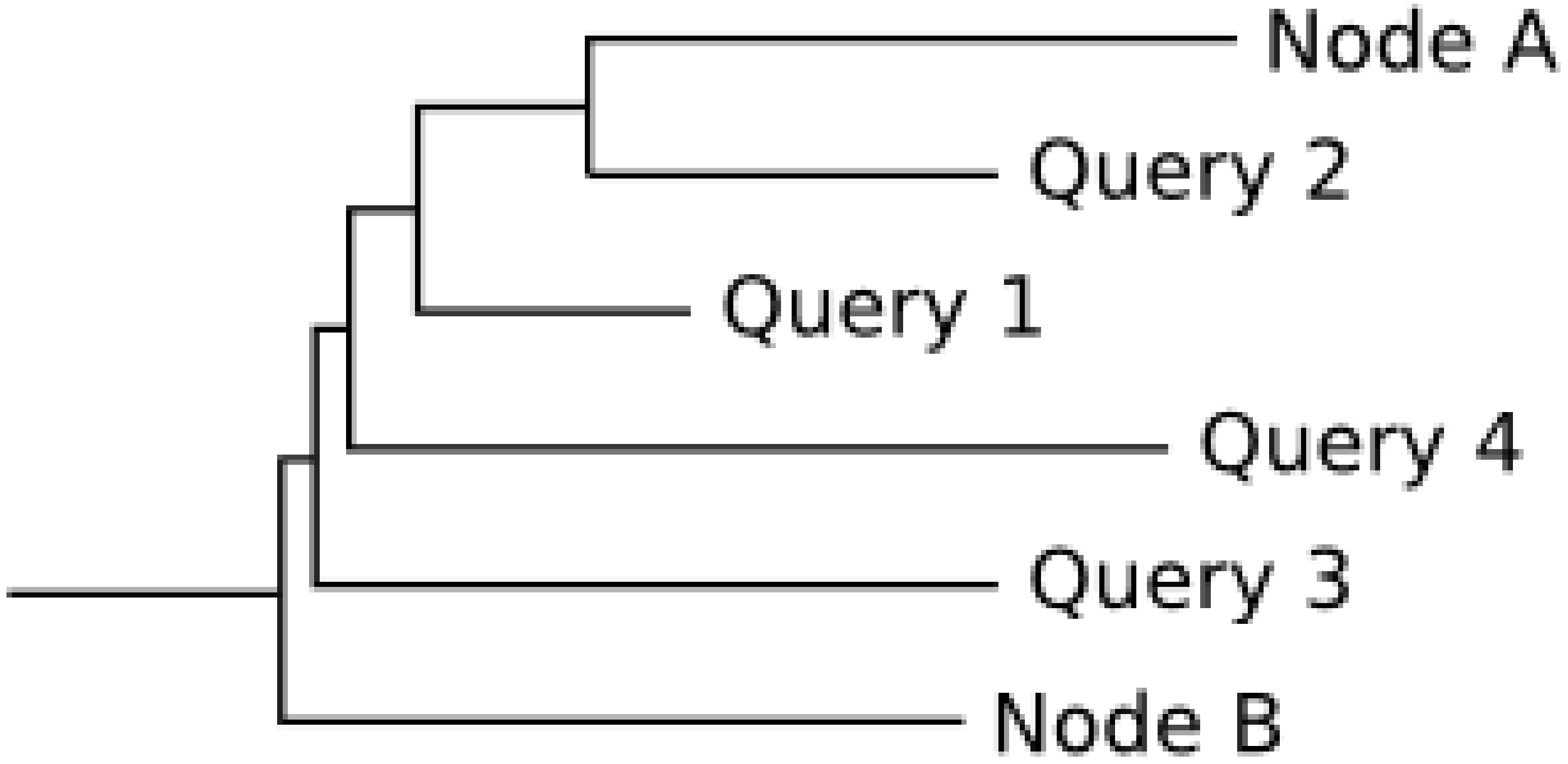
```
extract.sh.txt
extract_tree.svg
results.log
samples
sequences
```

Output

- **Color tree -** shows which branches of the tree were assigned to which clade
- Directory with jplace and fasta files
  - **basal_branches.jplace/fasta -** all placements that have their mass on branches that do not belong to any clade
  - **group1.jplace/fasta** - all placements that were placed in this group and that have passed likelihood weight threshold
  - **group2.jplace/fasta** - all placements that were placed in this group and that have passed likelihood weight threshold
  - **uncertain.jplace/fasta** - placements where no clade (including the basal clade) have more than the threshold amount of the mass in them

Labelled Tree

examine graft command creates a tree with each of the query sequences attached



More here: https://github.com/lczech/gappa/wiki/Subcommand:-graft

Workflow: labelled tree

COMMAND:

```
gappa examine graft --jplace-path epa_result.jplace --fully-resolve --out-dir labelled_tree
```

OUTPUT:

```
epa_result.newick
```

Placement Histograms

To check accuracy of the placement, we can use two types of histograms:

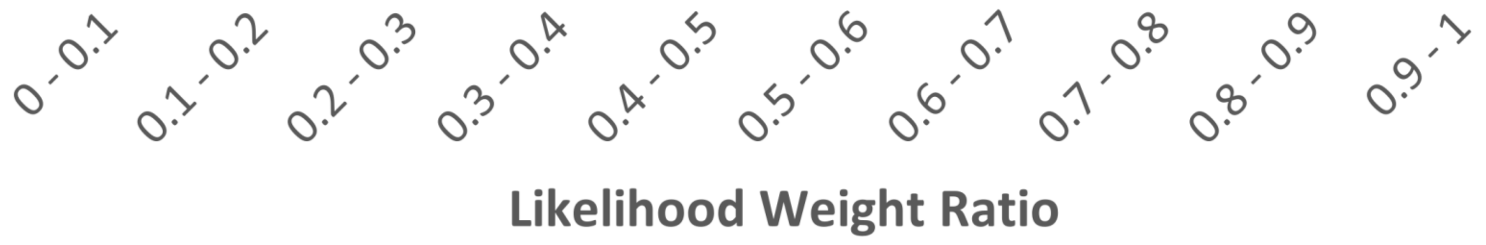
1) **LWR histogram** of the likelihood weight ratios (LWRs) of all placed OTUs

2) **EDPL histogram** of the expected distance between placement locations

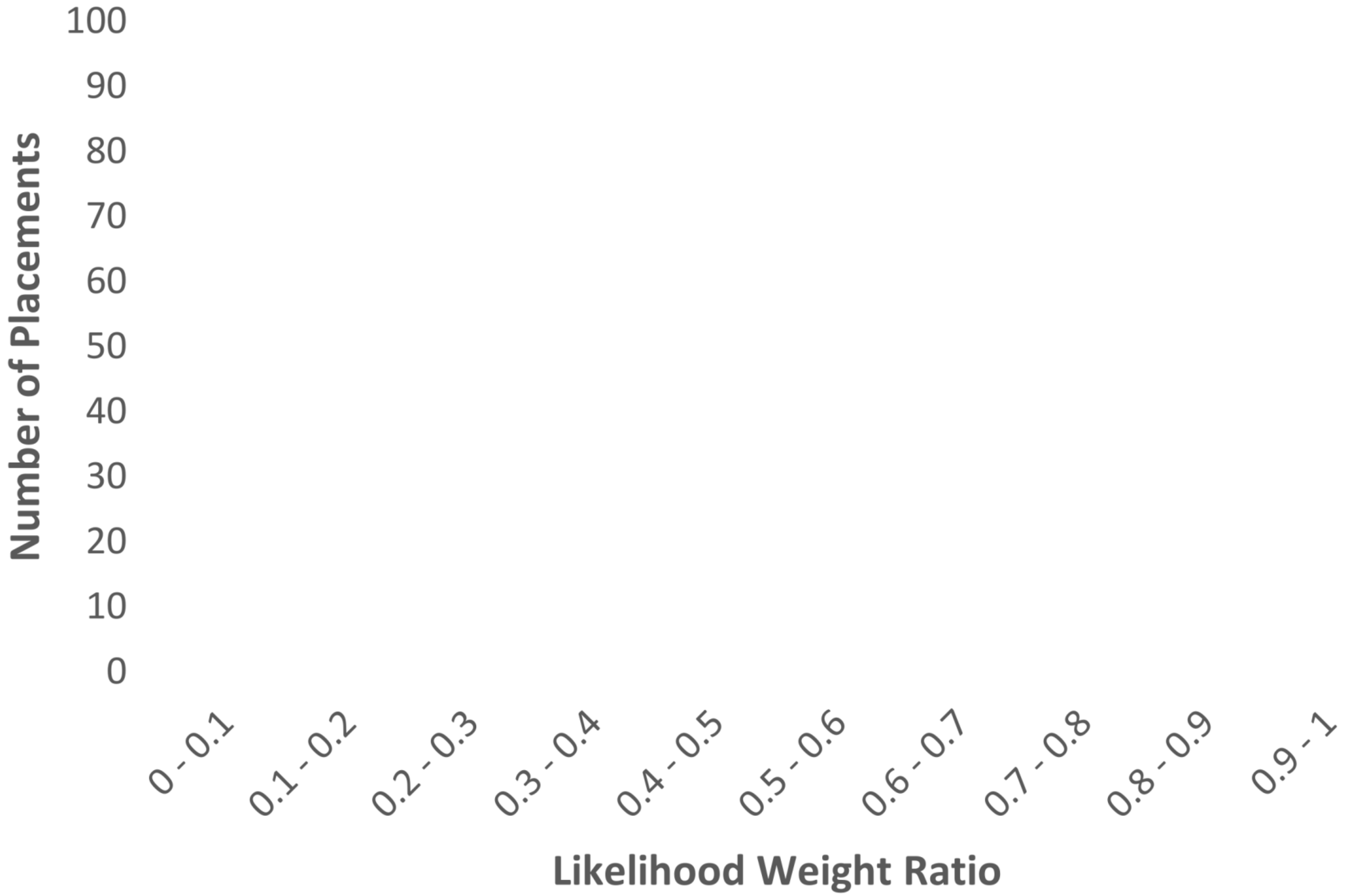
http://doc.genesis-lib.org/demos_placement_histograms.html

LWR histogram – accuracy of placements

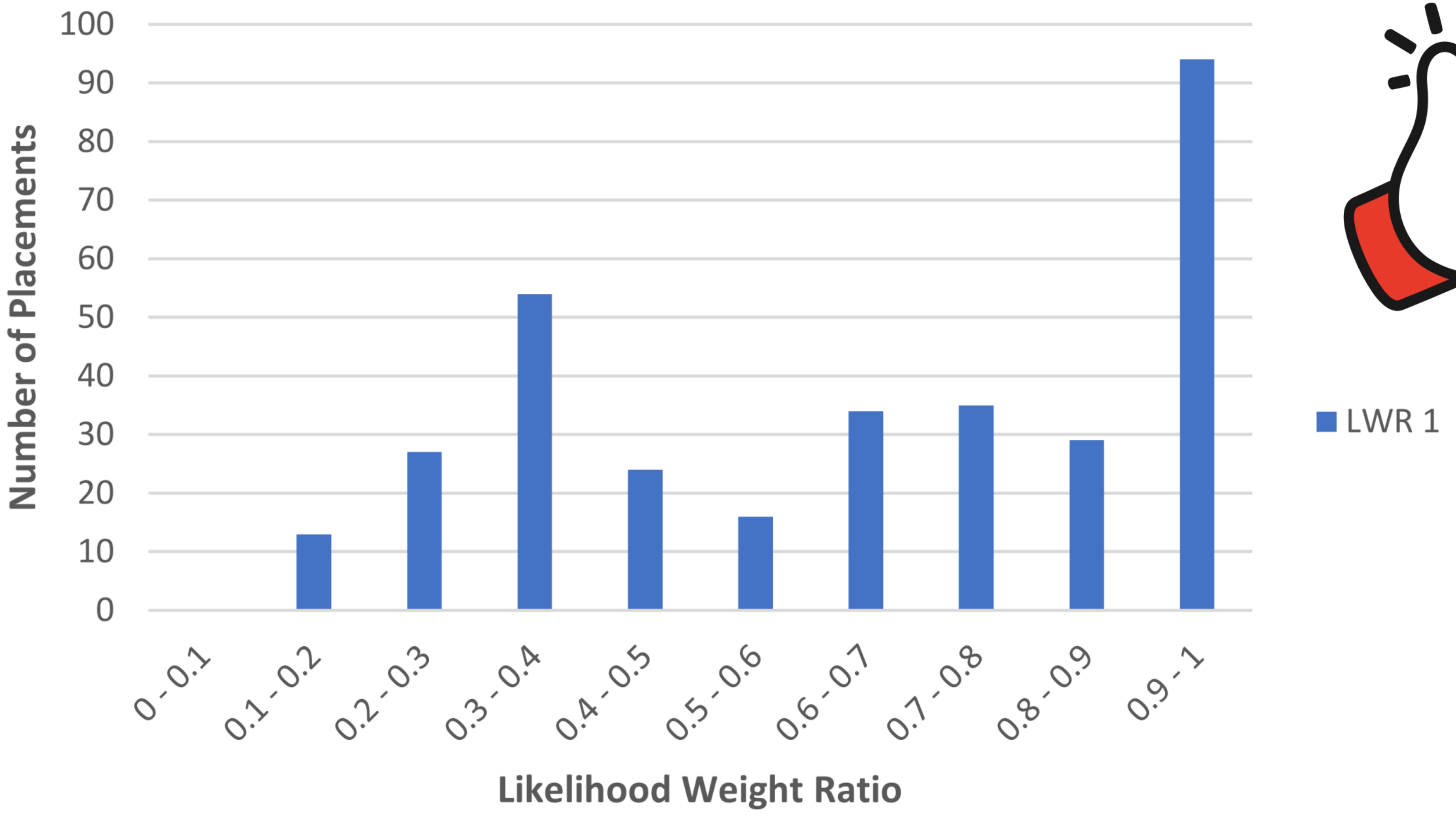
LWR histogram – accuracy of placements



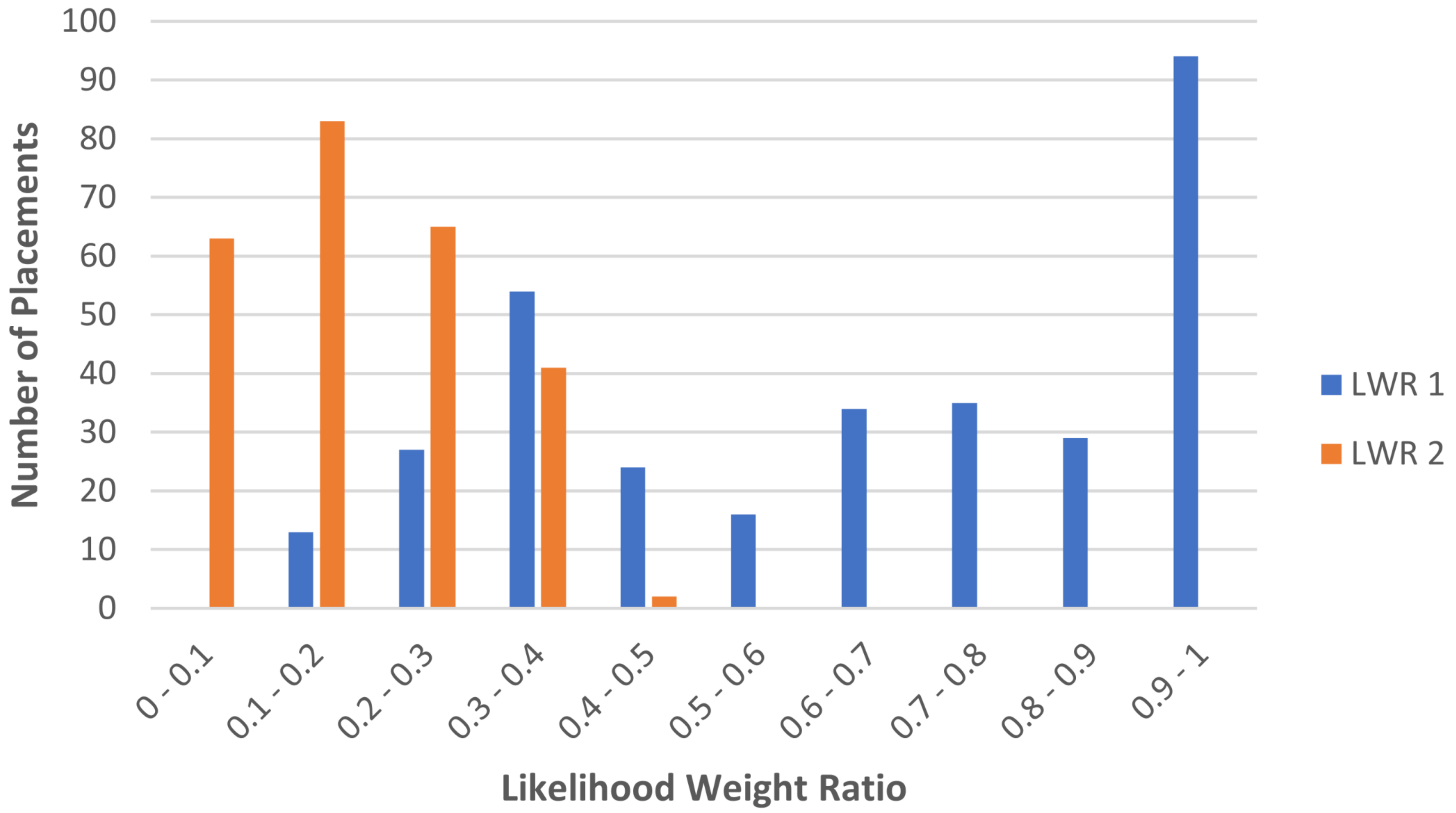
LWR histogram – accuracy of placements



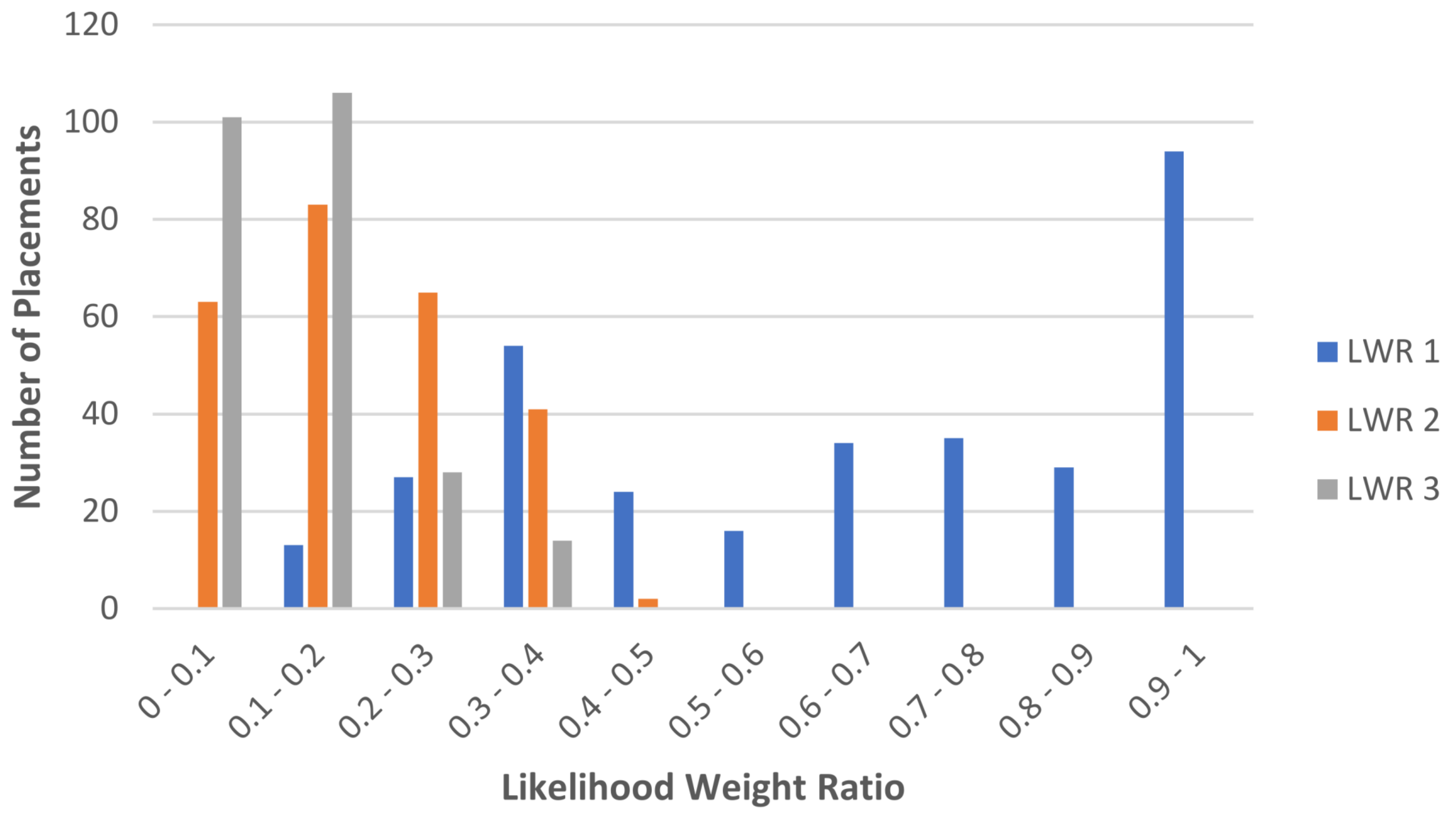
LWR histogram – accuracy of placements



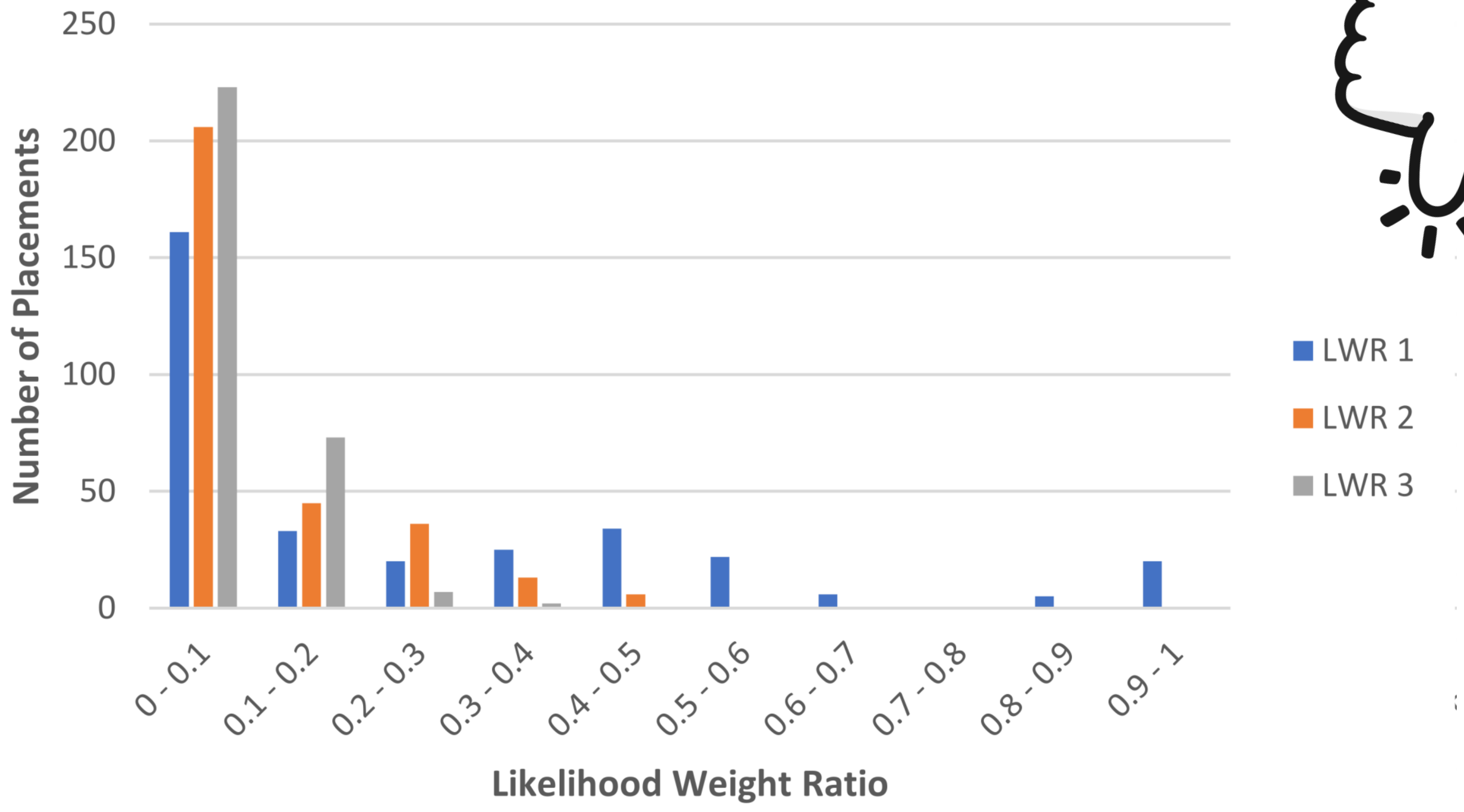
LWR histogram – accuracy of placements



LWR histogram – accuracy of placements

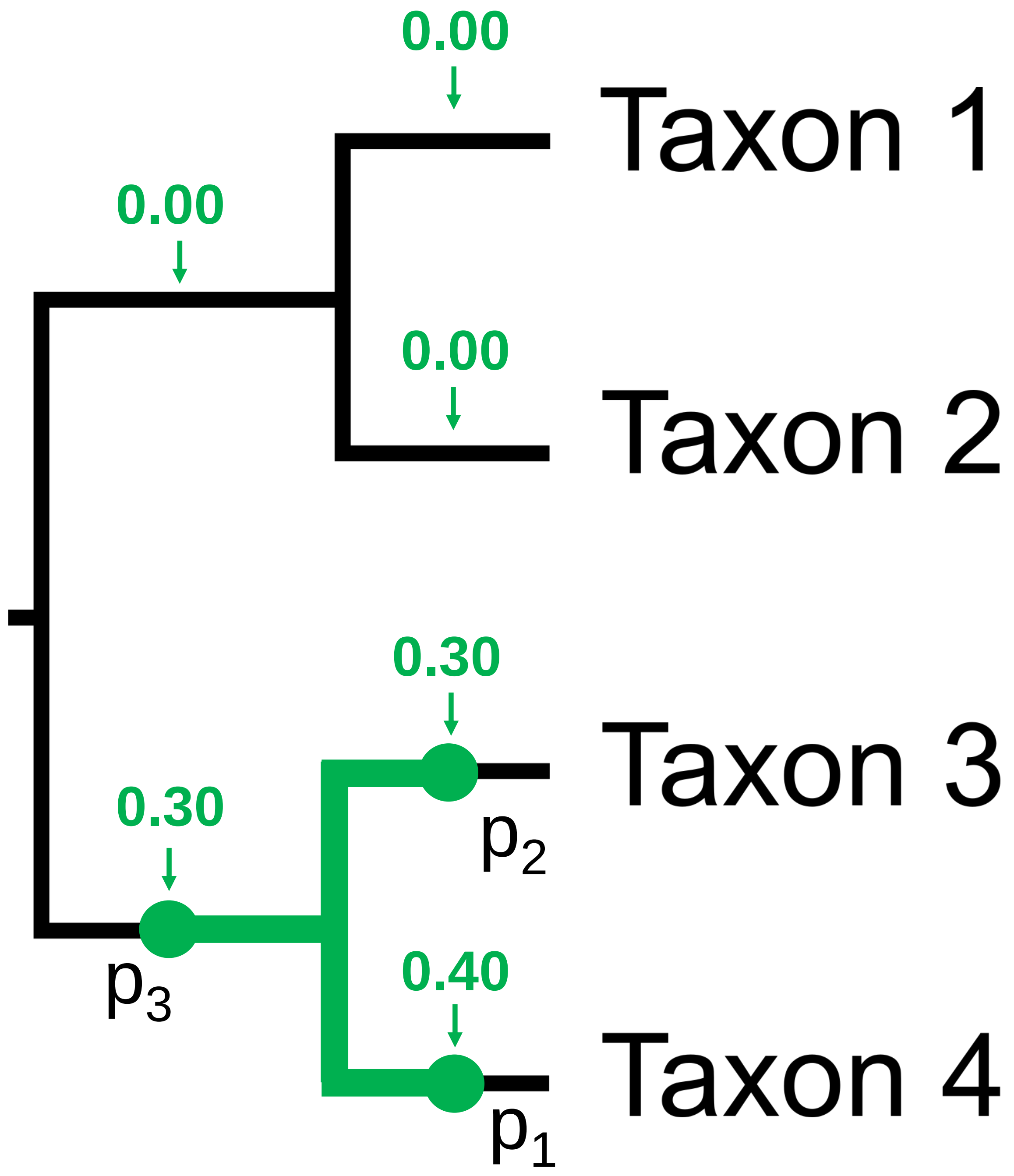


LWR histogram – accuracy of placements



But what if…

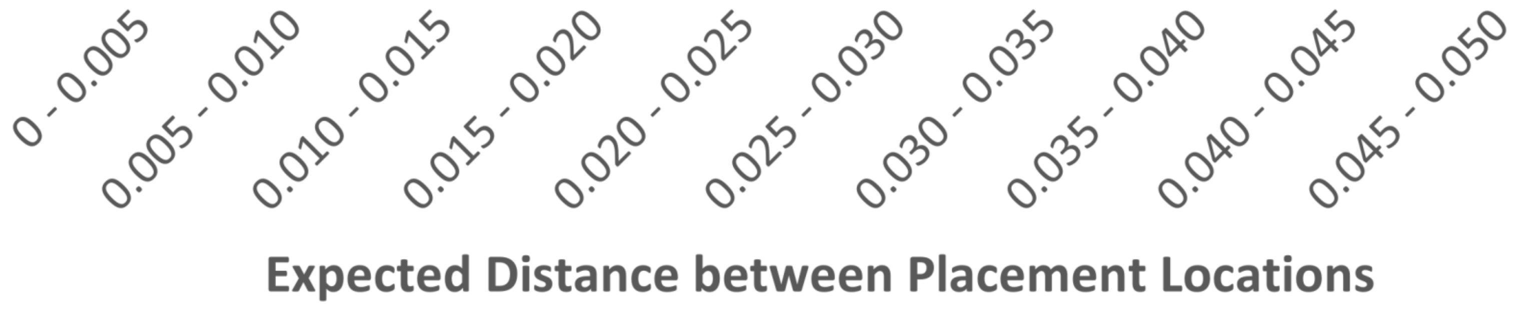
OTU_1 TGAAACAGTAAT



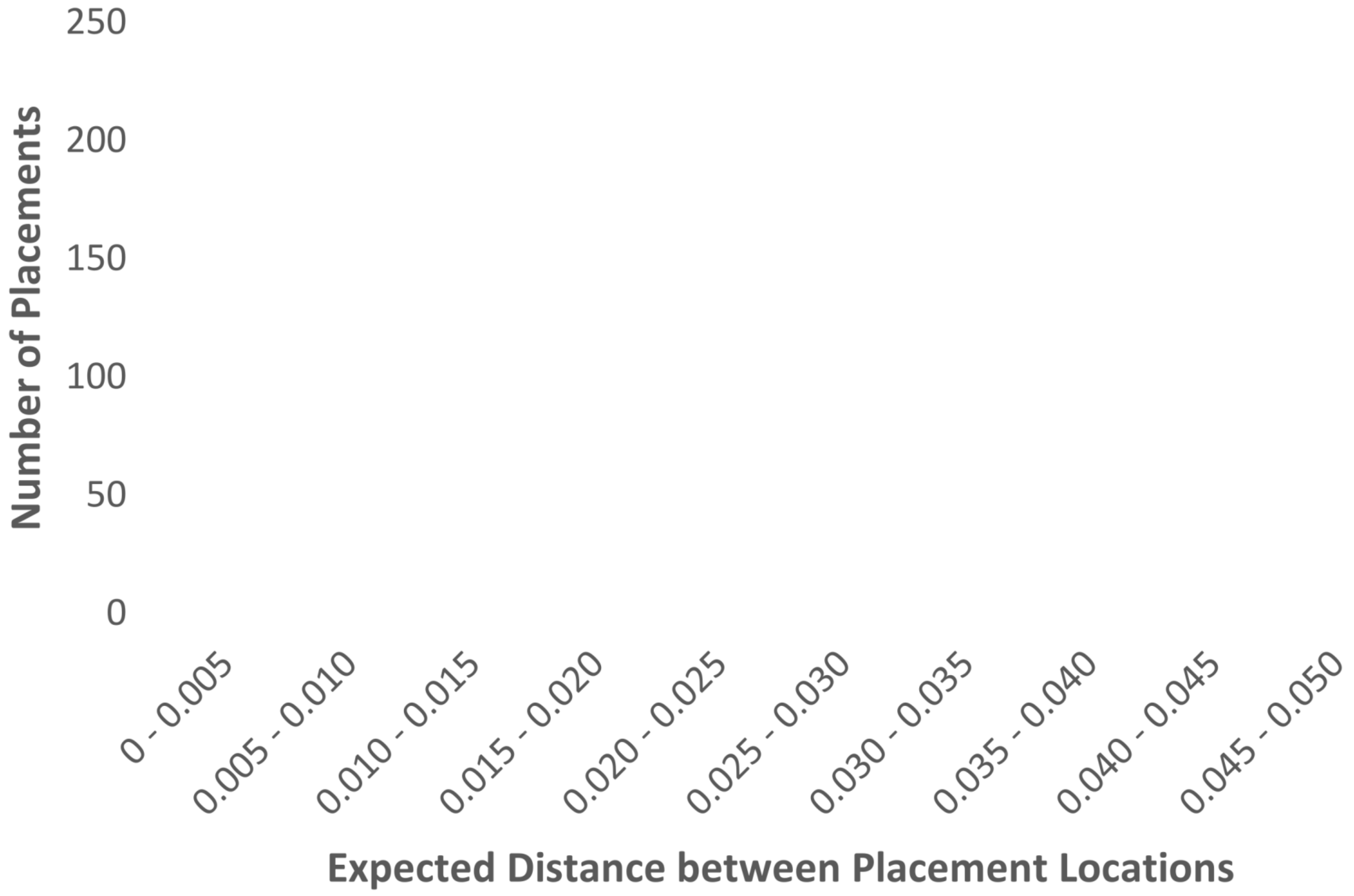
$$\boxed{\text{EDPL} = 2(p_1\, p_2\, d| + p_1\, p_3\, d| + p_2\, p_3\, d)}$$

EDPL histogram – distance between placements

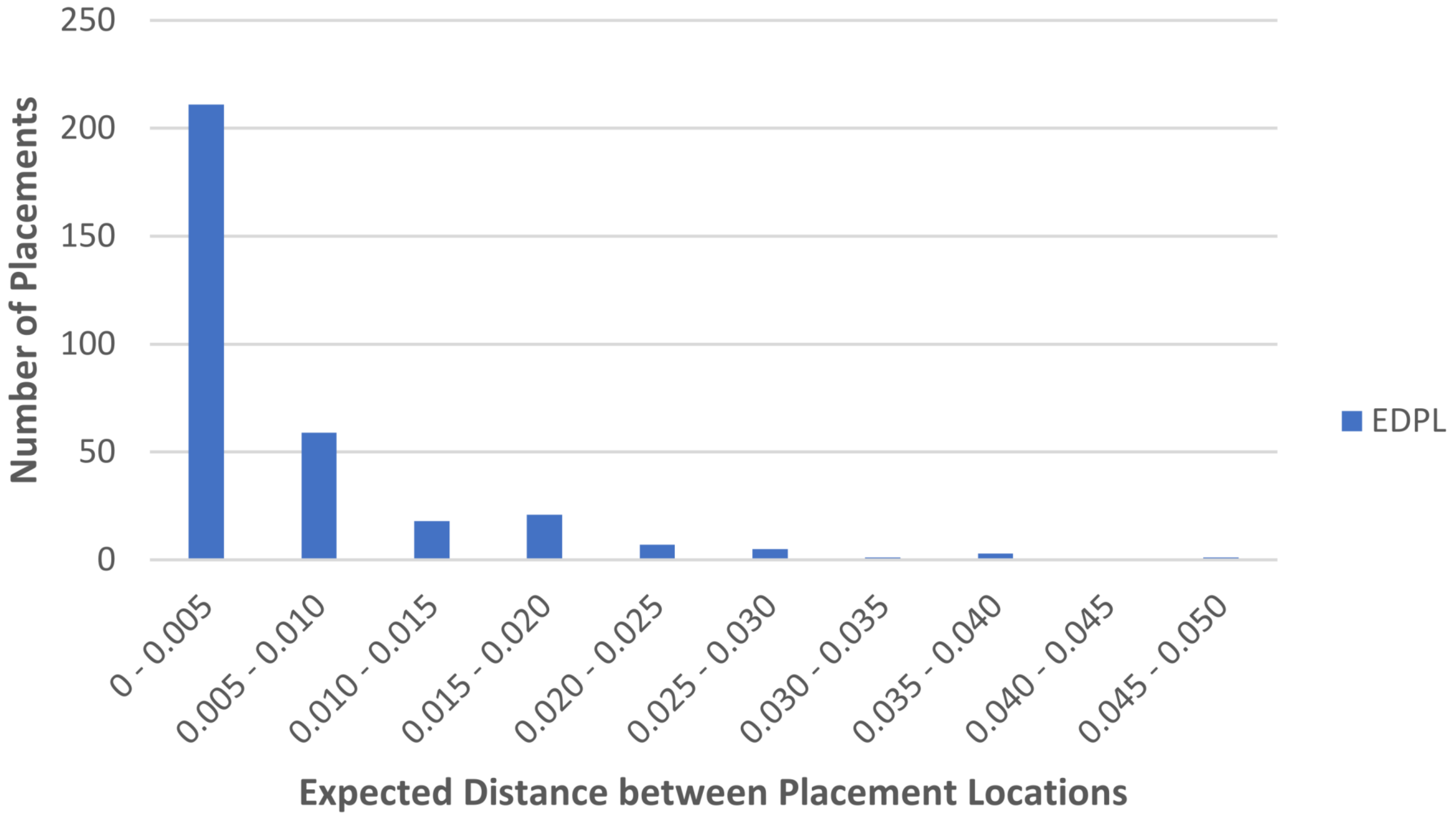
EDPL histogram – distance between placements



EDPL histogram – distance between placements



EDPL histogram – distance between placements



Workflow: histograms

LWR HISTOGRAM:

```
gappa examine lwr --jplace-path epa_result.jplace
```

OUTPUT:

```
lwr_histogram.csv
lwr_list.csv
```

EDPL HISTOGRAM

```
gappa examine edpl --jplace-path epa_result.jplace
```

OUTPUT:

```
edpl_histogram.csv
edpl_list.csv
```

Output Files

list.csv: A list of the LWRs and EDPL for each placed query of each sample.

histogram.csv: A summary histogram of the LWR and EDPL values.

This can be used in spreadsheet tools to produce a graph that allows an overview of the values for easy assessment.

https://github.com/lczech/gappa/wiki/Subcommand:-lwr

https://github.com/lczech/gappa/wiki/Subcommand:-edpl