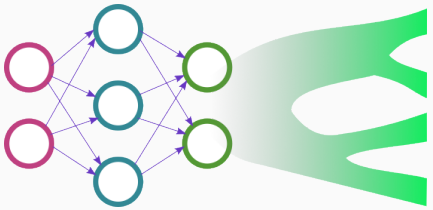
Deep likelihood-free inference of phylogenetic trees

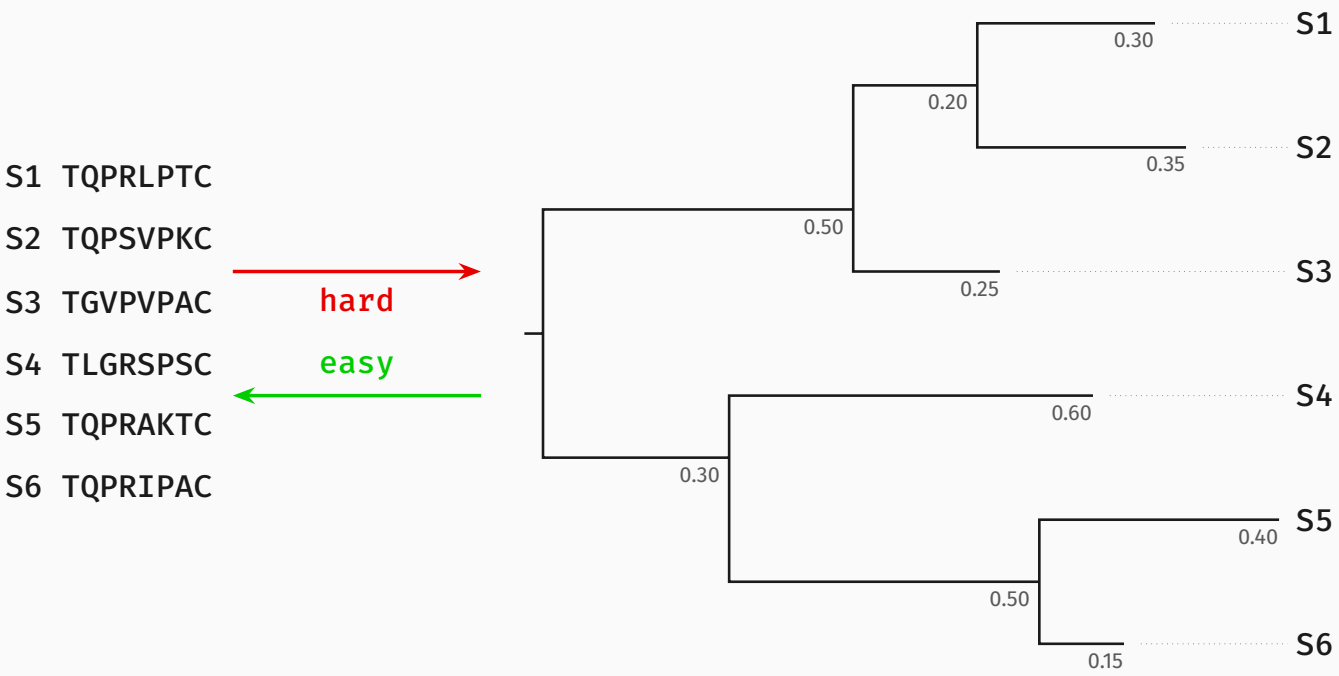


Luc Blassel, Nicolas Lartillot, Bastien Boussau, Laurent Jacob

DEELOGENY Meeting - June 13th, 2025

LBBE cnrs SORBONNE UNIVERSITÉ

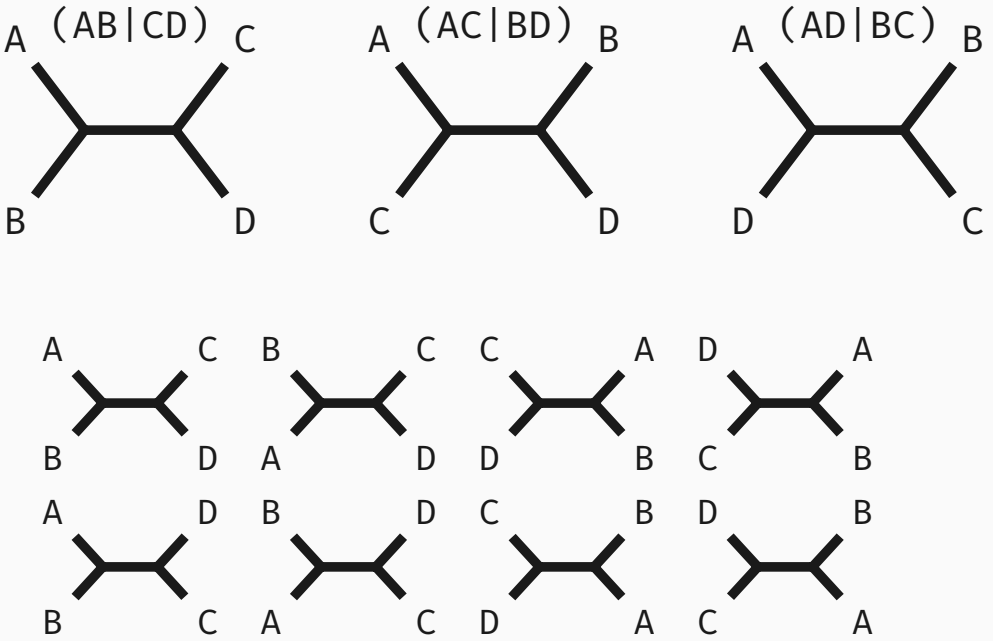
Motivation - Likelihood-free inference



- We can simulate many[1] (tree, MSA) pairs
- Can we **learn** the mapping **from MSA to tree**?

[1] effectively ∞

Intro - Quartet methods

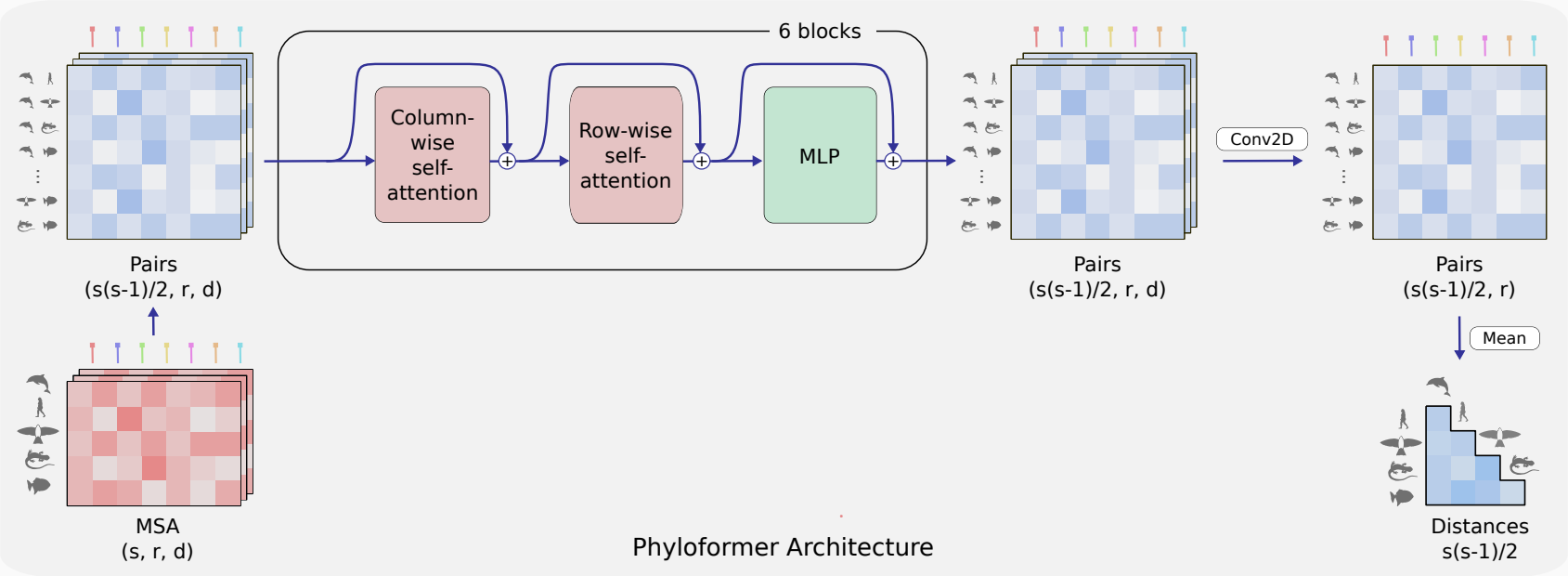


adapted from Tang et al. 2024

- **3** unique **topologies**
- Reframe as **classification** problem
- **Poor scalability:** $\mathcal{O}(n^4)$
- **Poor performance** in some settings

Tang et al. 2024; Suvorov et al. 2019; Zou et al. 2020

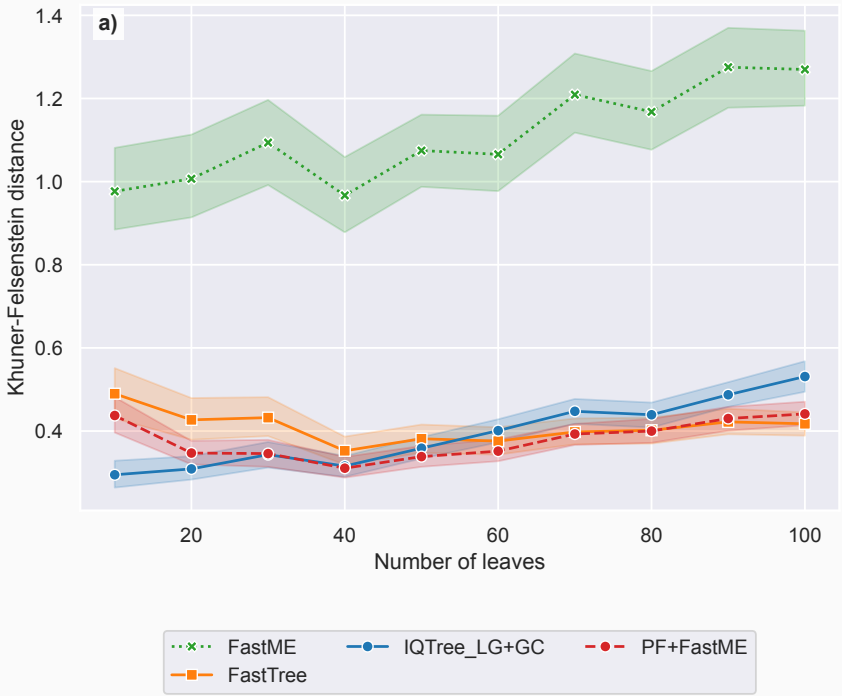
Intro - Phyloformer, our first approach

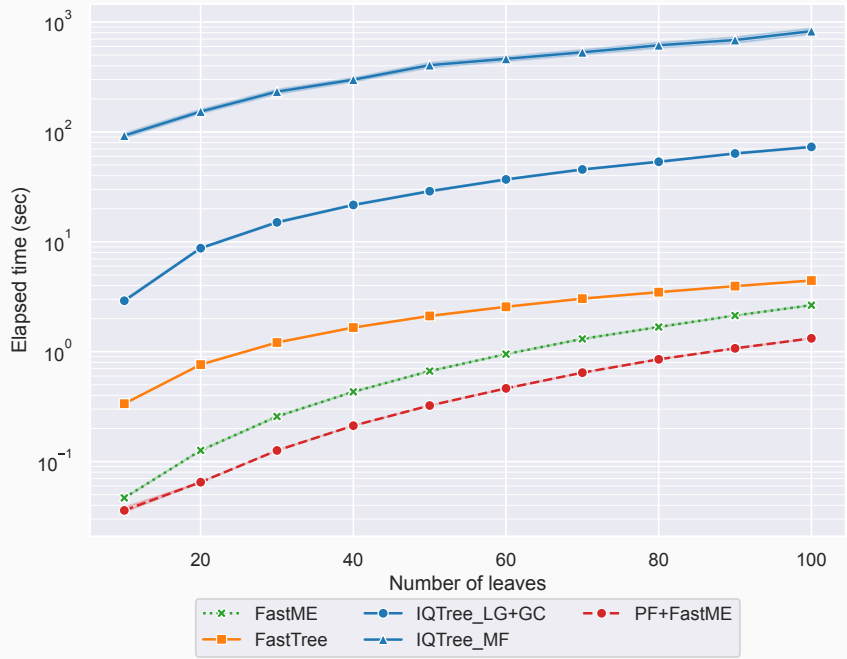


- Input an **MSA**, get a **Distance matrix**
- Feed Distance matrix to **FastME** to get **tree**

Nesterenko et al. 2025; Lefort et al. 2015

Intro - Phyloformer is good!



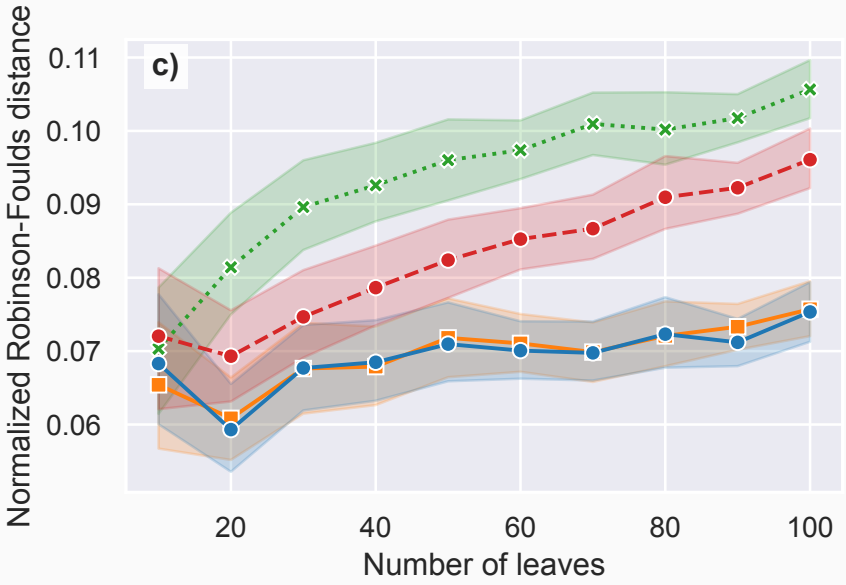


Tree inference accuracy (KF) *Runtime*

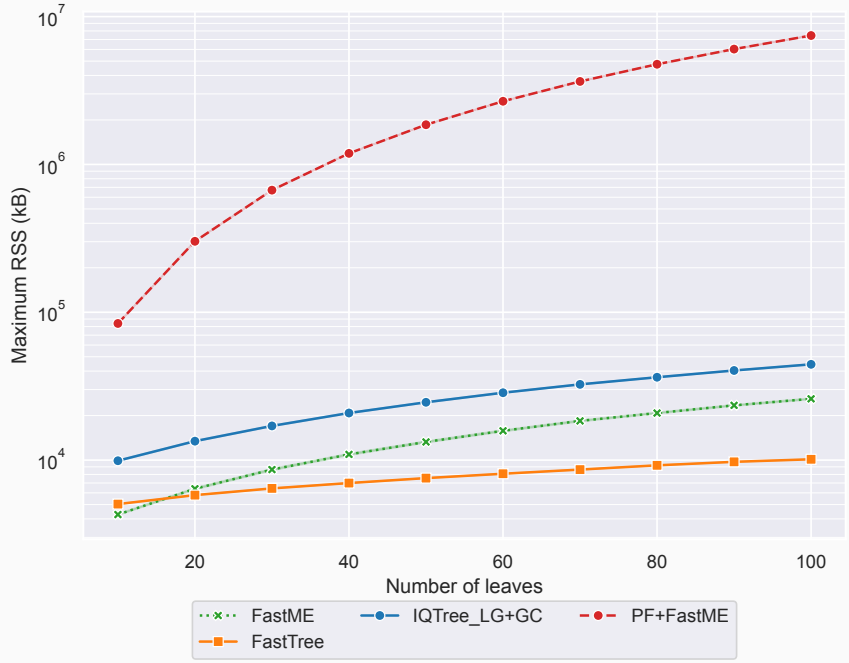
- Fairly **competitive** even on simple LG+GC model
- **Fast** because we use GPUs [1]

Nesterenko et al. 2025, [1] 🙏 Jean-Zay

Intro - But Phyloformer is less good...



Topological accuracy (RF)



Memory usage

- **Gap** between PF and **ML methods**
- PF is **by far** the most **memory intensive**

How to do phylogenetic inference end-to-end ?

Methods - Initial attempts a differentiable NJ

- With Phyloformer we **predict distances**, we need **tree-building** algorithm afterwards
- Can we just **add NJ** to the end of Phyloformer and **learn through it**?[1]

NJ applied to D:

1. Compute $Q = f(D)$
2. Find $(i, j) = \underset{i \neq j}{\text{argmin}}(Q_{ij})$
3. Merge (i, j) into u, compute δ_{iu}, δ_{ju}
4. Update $D_{uk} \forall (k \notin (i, j))$
5. Go back to 1. until topology is resolved

[1] Spoilers, not really ...

Methods - Making NJ differentiable

- The first **problem** is the **discrete** argmin:
  - We can use **perturbations** to **approximate** ∇_θ
  - Gumbel-softmax **straight-through** trick:

$$O = \text{armgin and } \mathcal{O} = \text{softmin}$$
$$\text{forward:} \quad x \mapsto O(x)$$
$$\text{backward:} \quad \nabla_\theta O(x) \approx \nabla_\theta \mathcal{O}(x)$$

- **What loss** can we use ?
  - Approximate RF: **unstable**
  - Optimal Transport: did **not** manage make **work**
  - Distances: **not great** performance

In summary: **Not the right approach**[1]

Berthet et al. 2020; Jang et al. 2017

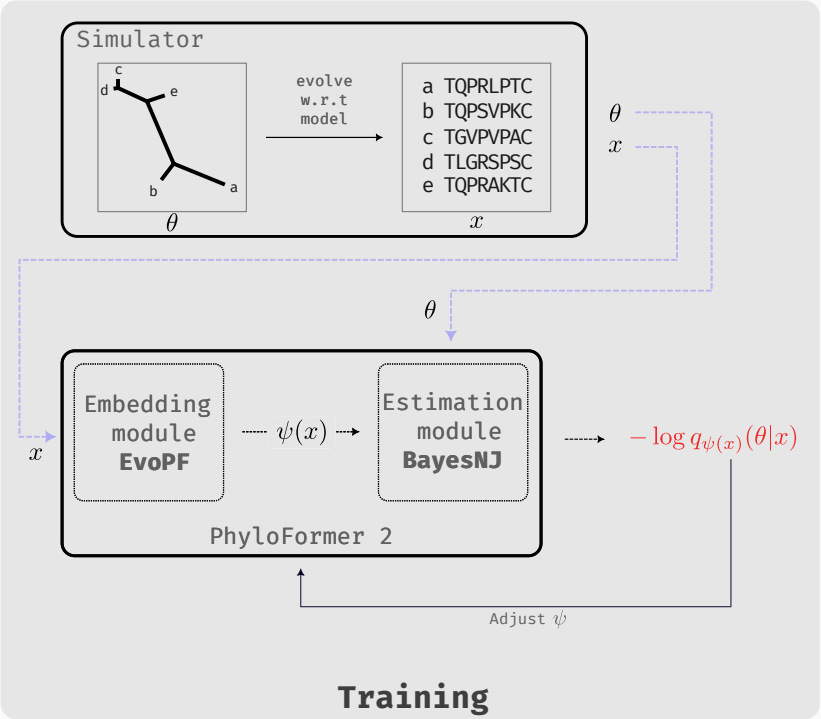
[1] But cool nonetheless

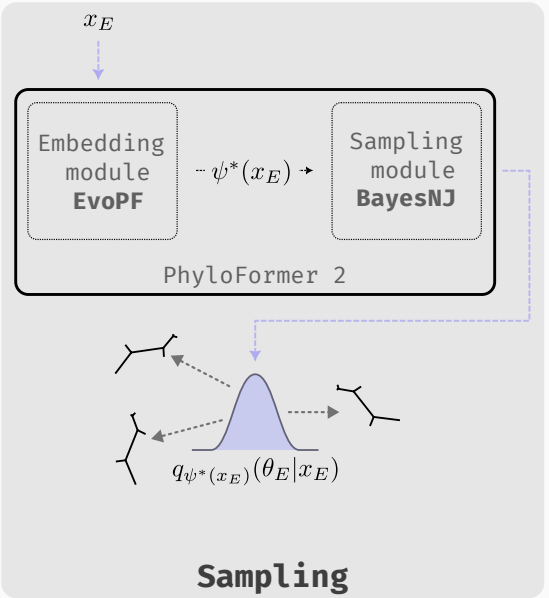
Methods - Neural Posterior Estimation (NPE)

- Given a **probabilistic model** $p(x|\theta)$ with some prior $p(\theta)$
- We want to **estimate the posterior**: $p(\theta|x)$
- We build $q_\psi(\theta|x)$ a **family** of distributions **parametrized** by ψ (our NN)
- We find $q_{\psi^*} = \underset{\psi}{\text{argmin}}\ \mathbb{E}_{p(x)}[KL(q_\psi(\theta|x)||p(\theta|x)]$
- In practice we **minimize** $\mathbb{E}_{p(x,\theta)}[\log q_{\psi(x)}(\theta|x)]$ by **sampling** from $p(x, \theta)$

x : MSA, $\theta = (\tau, \ell)$: Phylogenetic tree, $\psi(x)$: NN applied to x

Methods - How do we do NPE?





- During **training** find $\psi^* = \underset{\psi}{\operatorname{argmin}} \; -\sum_i \log q_{\psi(x_i)}(\theta_i|x_i)$
- At **inference** time **sample** from: $q_{\psi^*(x_E)}(\theta_E|x_E)$

Methods - The EvoPF module, intro

the EvoPF module is an **adaptation** of the **EvoFormer** module from AlphaFold2. The tasks are **transpositions** of each other:

given input MSA ($n \times r$)

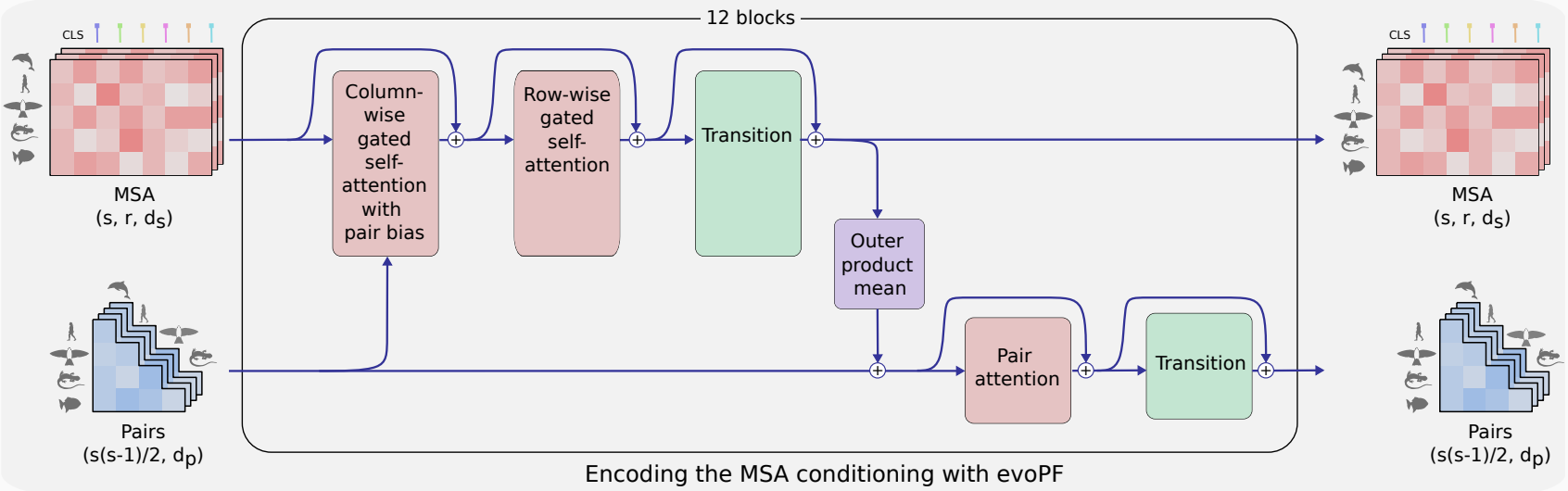
EvoFormer represent $r \times r$ relationships between sites

EvoPF represent $n \times n$ relationships between sequences

More expressive than MSA transformer
More lightweight than PF

Jumper et al. 2021; Rao et al. 2021

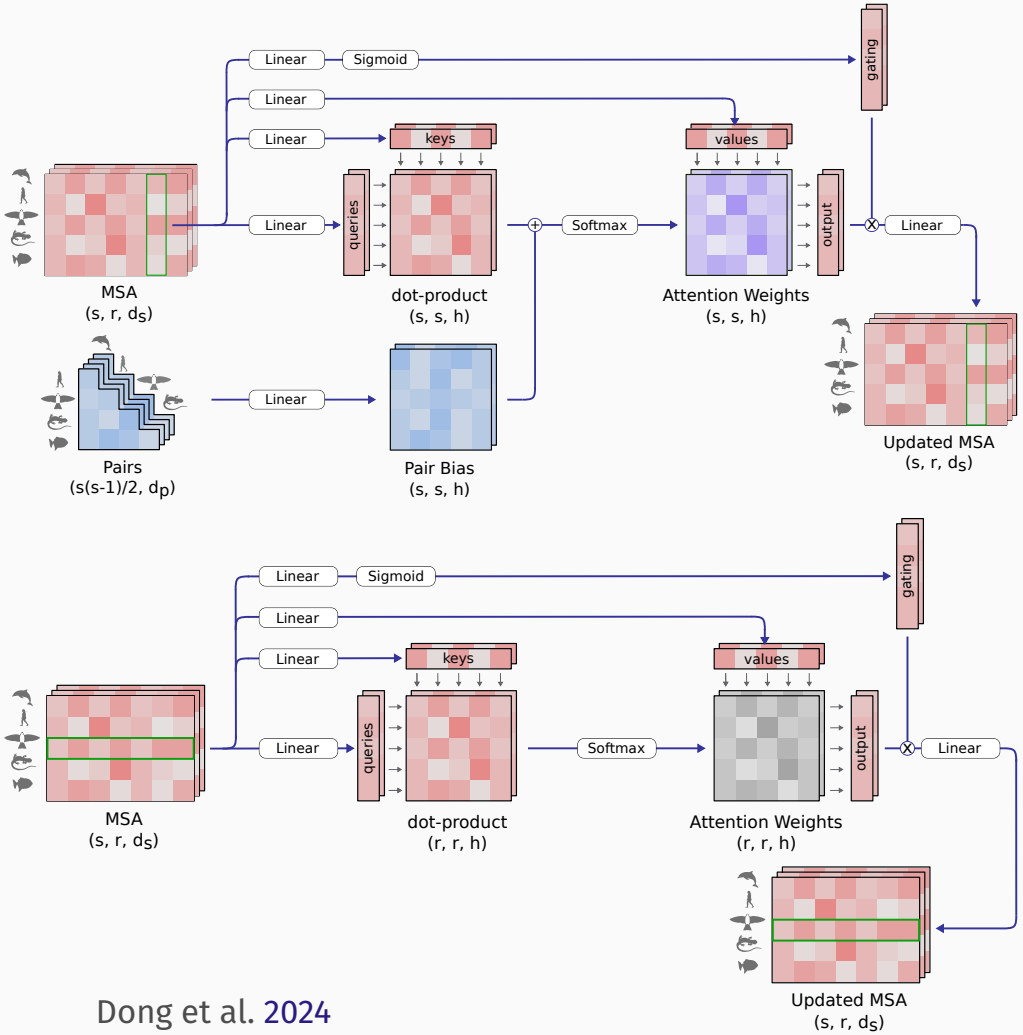
Methods - The EvoPF module, details



- Input an **MSA** and get:
  sequence embedding $\{s_i\}$
  sequence-pair embeddings $\{z_{ij}\}$
- **Both** embedding-types used to **update each-other**

Figure inspired by Jumper et al. 2021

Methods - EvoPF, the MSA stack

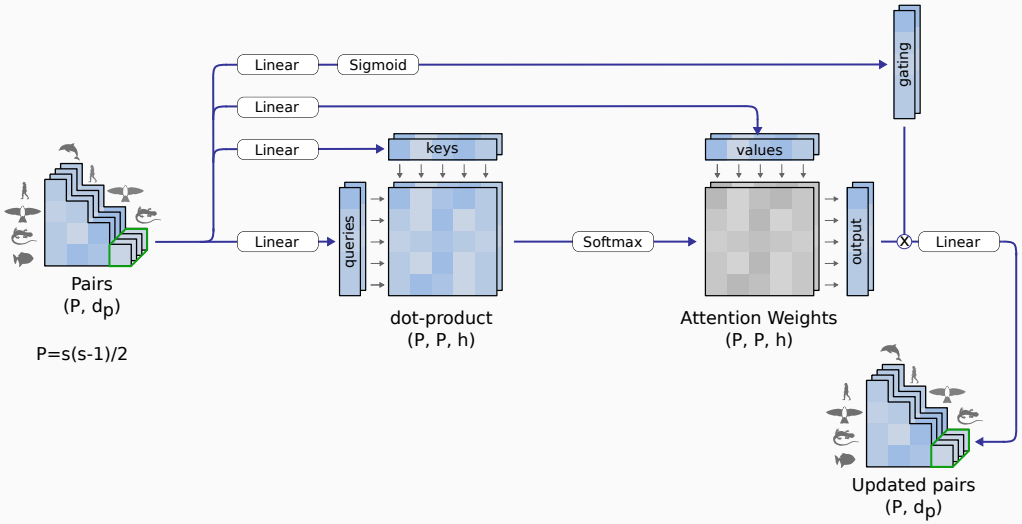


Dong et al. 2024

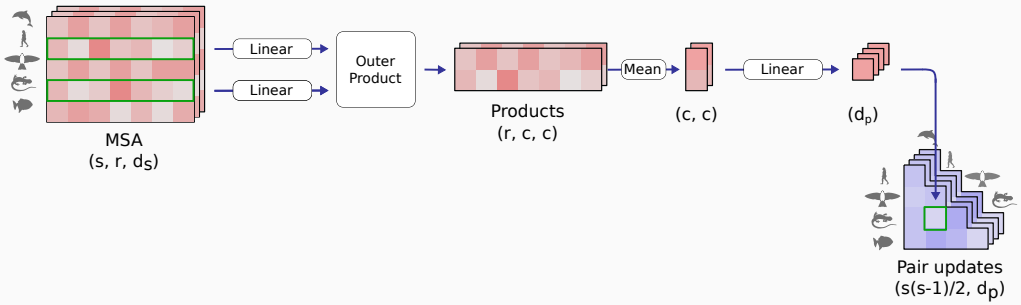
Column-wise attention with pair-bias

Row-wise attention

Methods - EvoPF, the pair stack



Pair attention

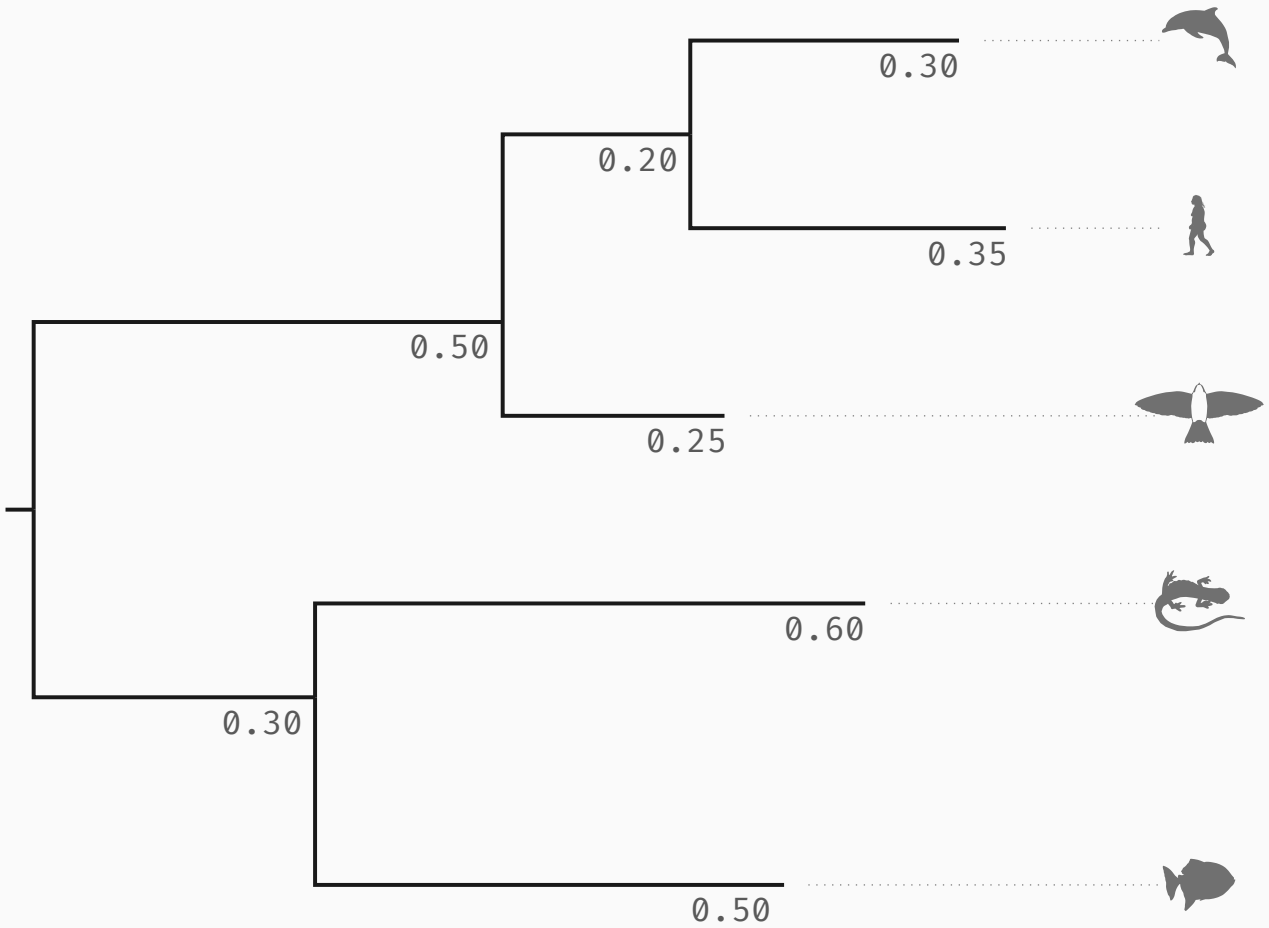


Outer product mean

Dong et al. 2024

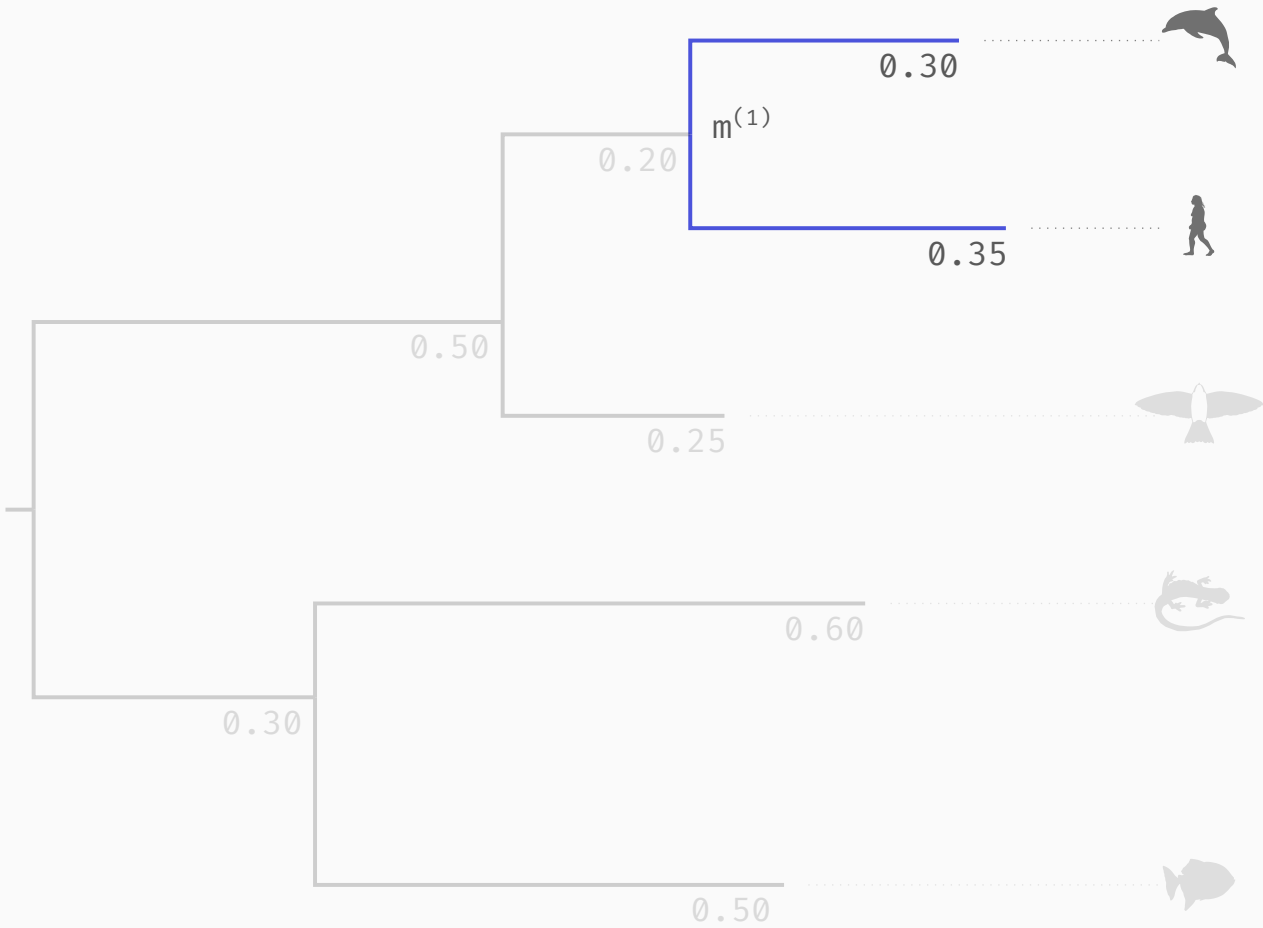
Methods - A tree is a series of merges

We want to describe the following tree:



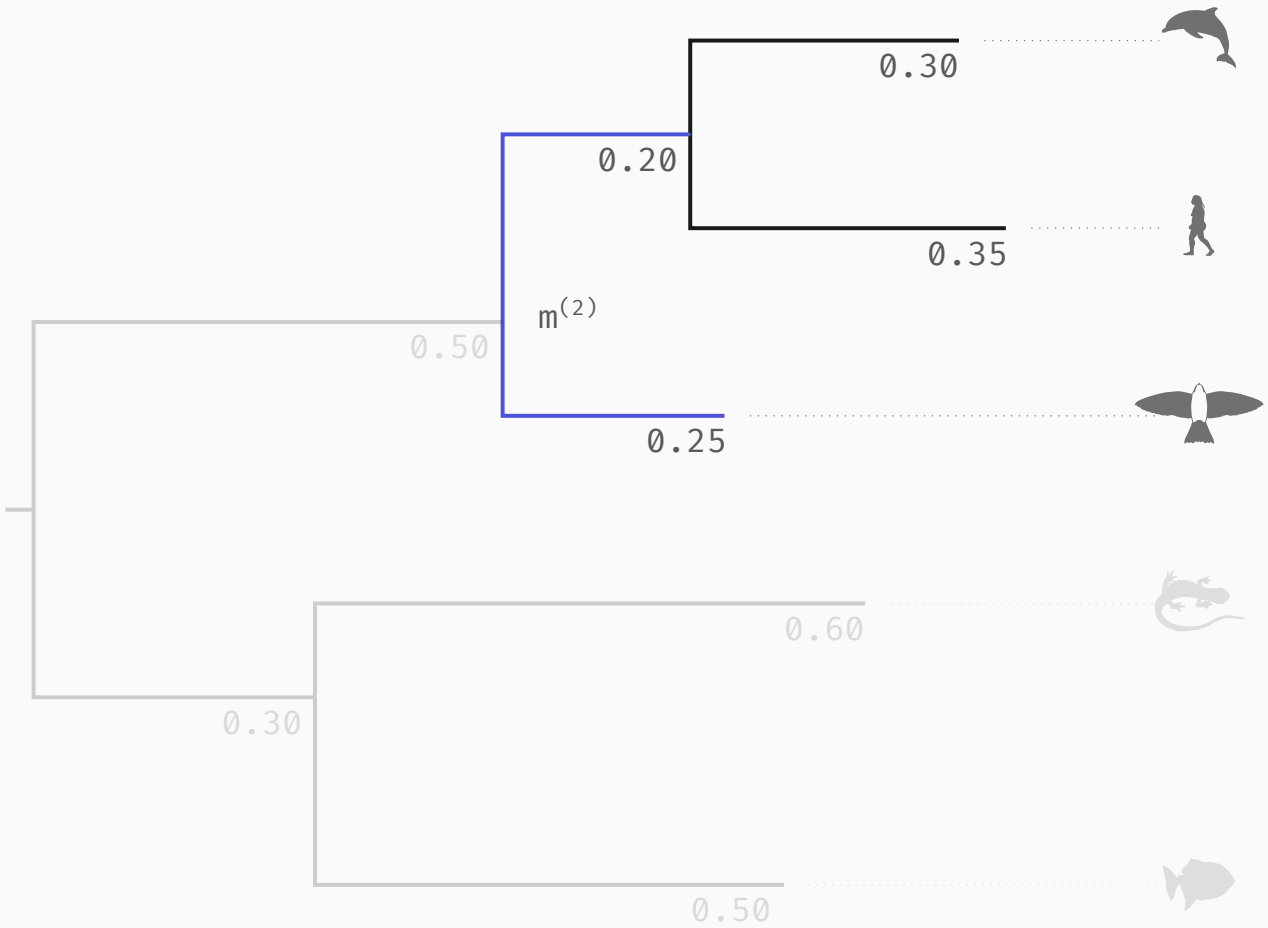
Methods - A tree is a series of merges

Iteratively create cherries:



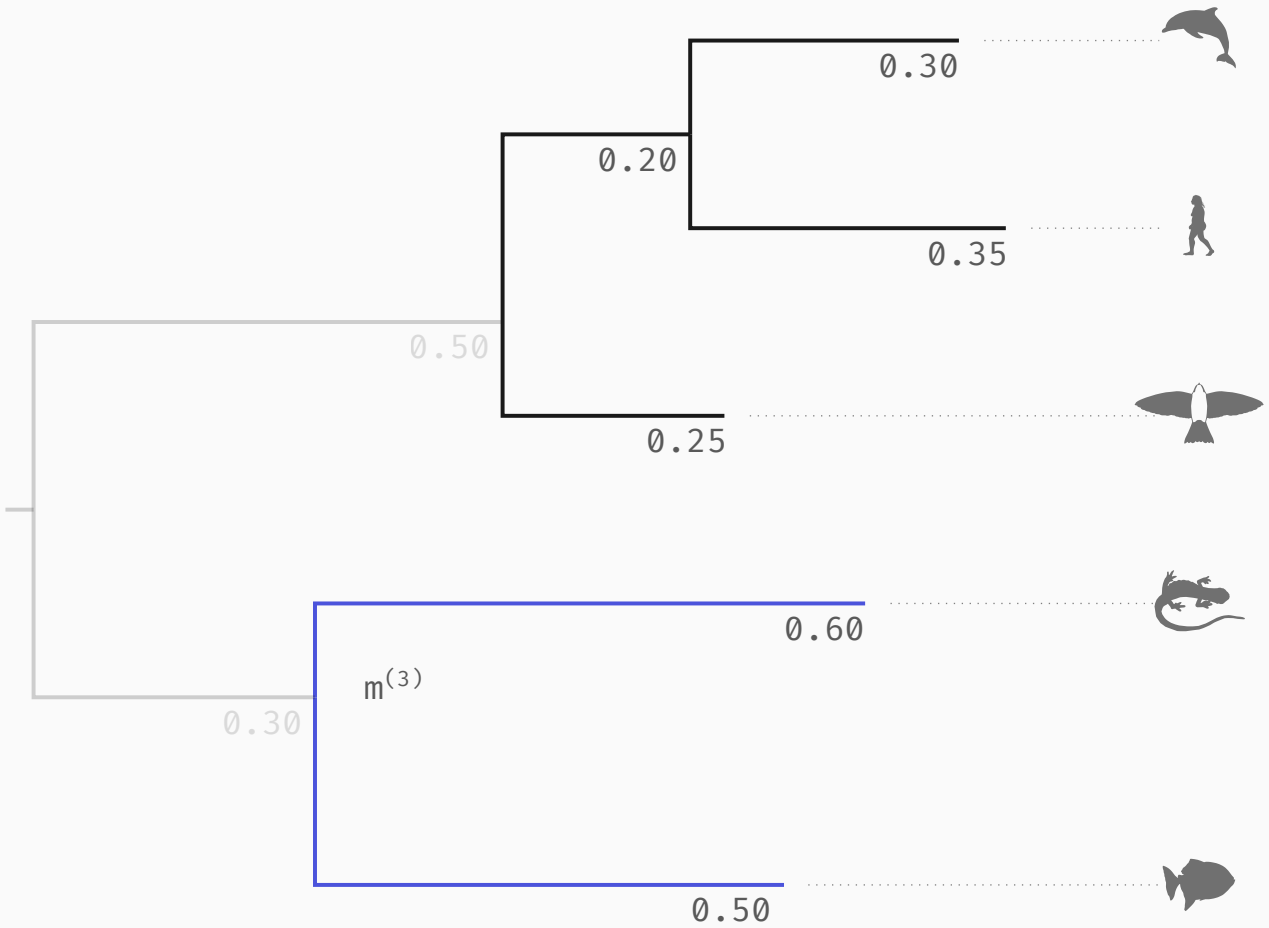
Methods - A tree is a series of merges

Iteratively create cherries:



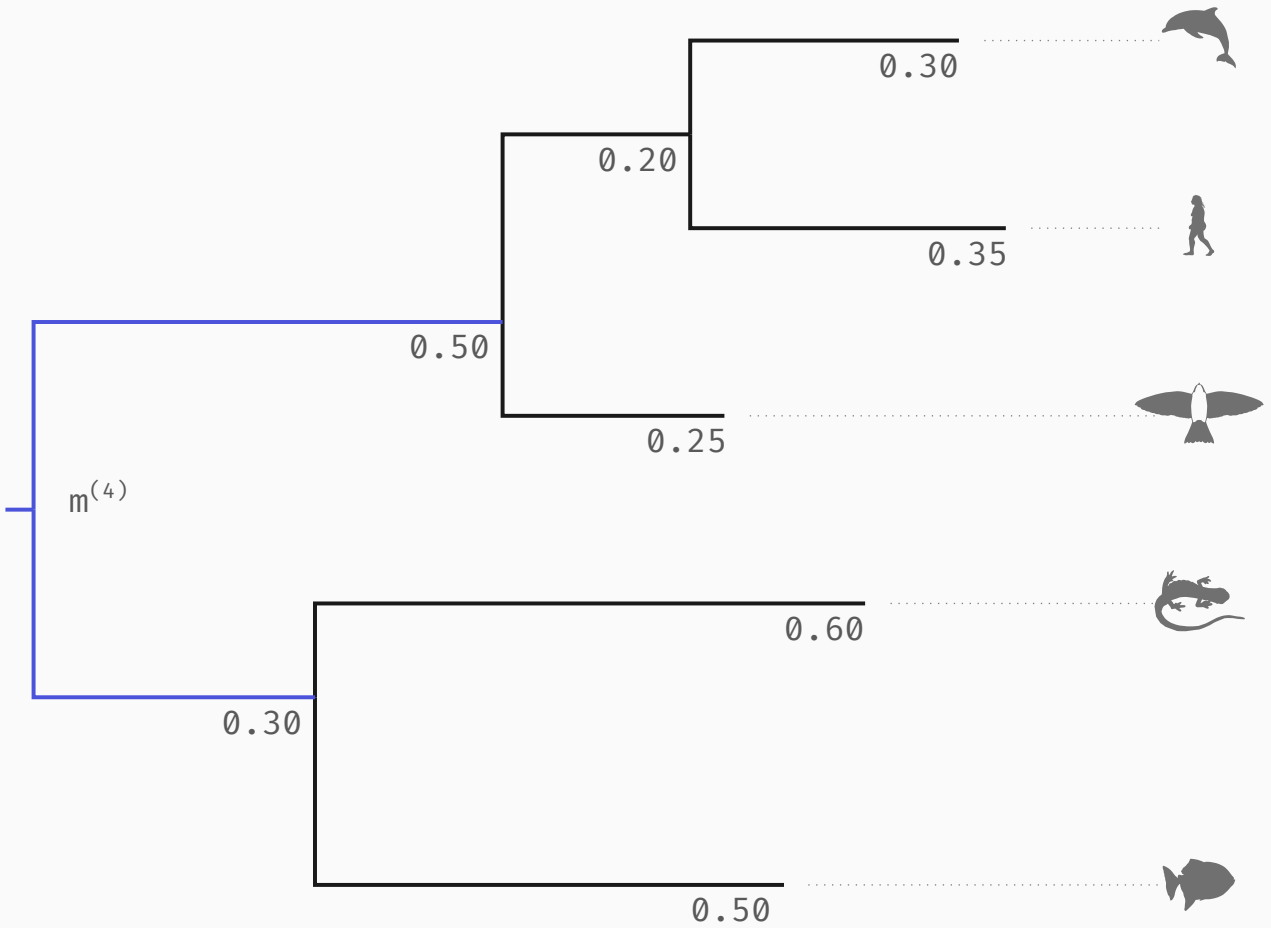
Methods - A tree is a series of merges

Iteratively create cherries:



Methods - A tree is a series of merges

Iteratively create cherries:



Methods - the BayesNJ module

- **Tree** is an **ordered set** of merges: $\theta : \{m^{(1)}, \ldots, m^{(N-1)}\}$
- We **factorize** $q_{\psi(x)}(\theta|x)$ as the product of successive merge probabilities:

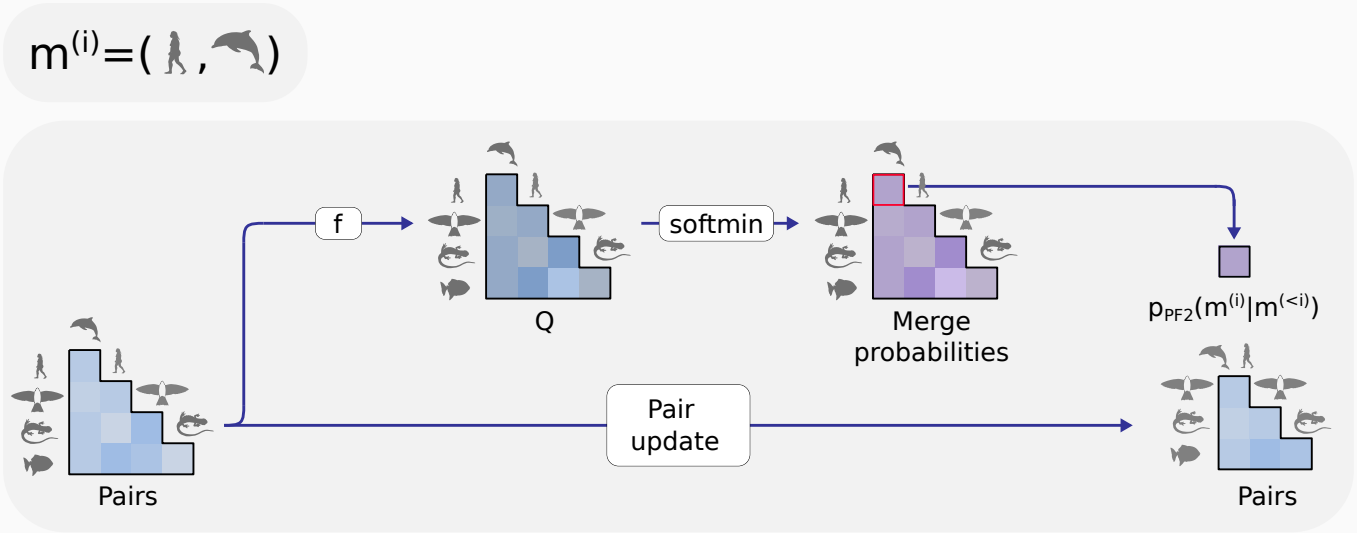
$$q_{\psi(x)}(\theta|x) = \prod_{k=1}^{N-1} q_m(m^{(k)}|m^{(<k)}) q_\ell(\ell^{(k)}|m^{(\leq k)})$$

- **Merge** probabilities have **2 components**:

topological: $q_m(m^{(k)}|m^{(<k)})$

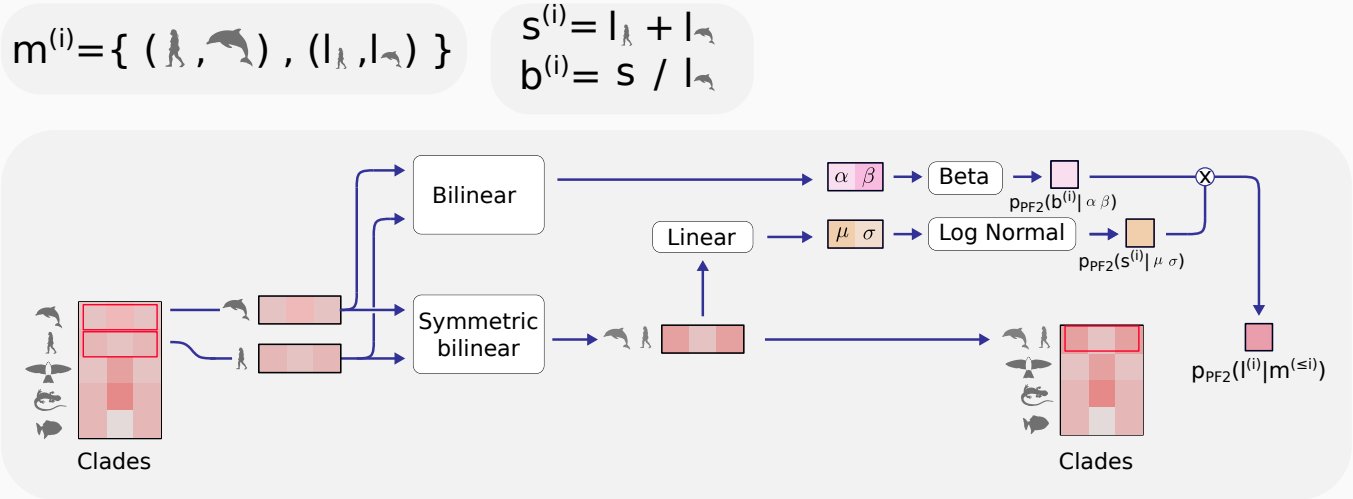
branch-length: $q_\ell(\ell^{(k)}|m^{(\leq k)})$

Methods - BayesNJ, evaluating topological probabilities



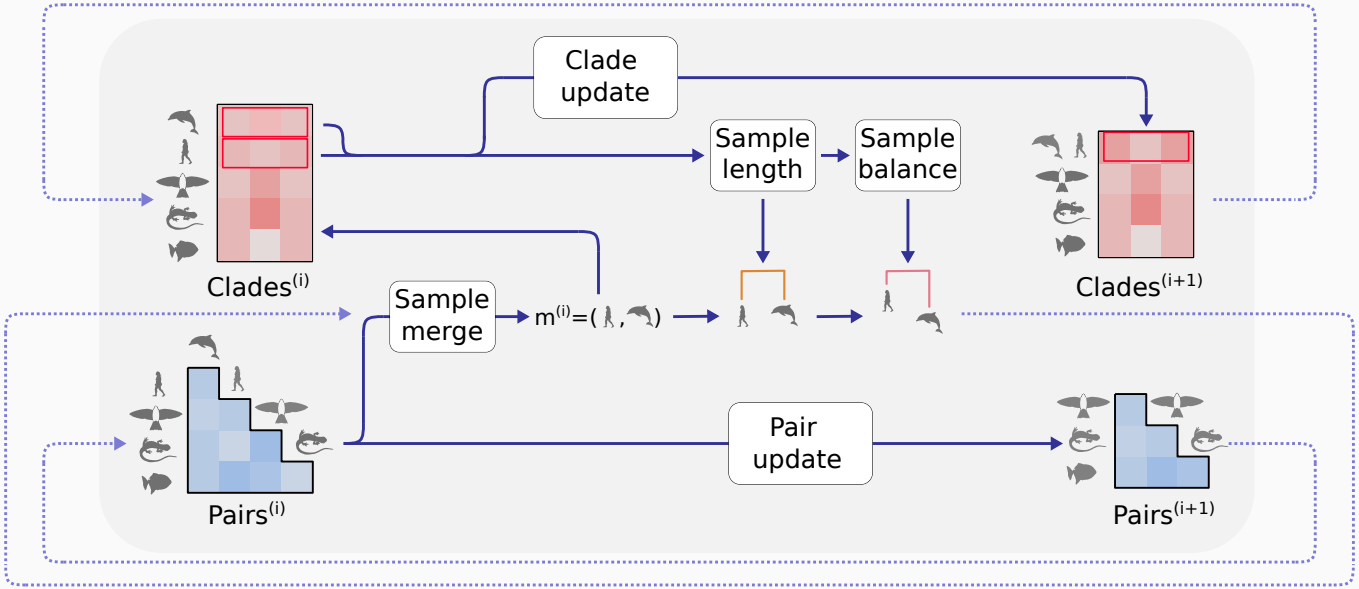
- **Adaptation** of **NJ** algorithm
- Treating **pair-embeddings** like **distances**

Methods - BayesNJ, evaluating branch length probabilities



- Compute **probability** of $s^{(i)}$ the **cherry length**
- Compute **probability** of $b^{(i)}$ the **cherry balance**
- **Combine** both for branch-lengths probability

Methods - BayesNJ sampling mode



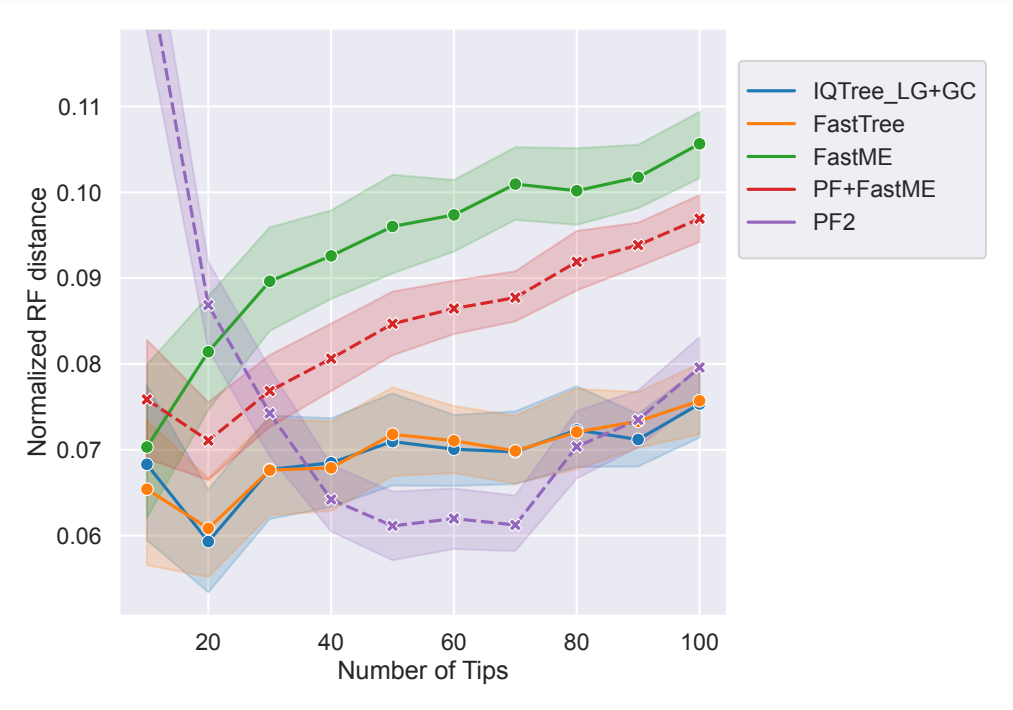
- **Sample** merges and branch lengths **until** topoloy **resolved**
- **Two** sampling **modes** given $\psi(x_E)$:

  Bayesian Sample from distributions

  Greedy MAP Choose mode

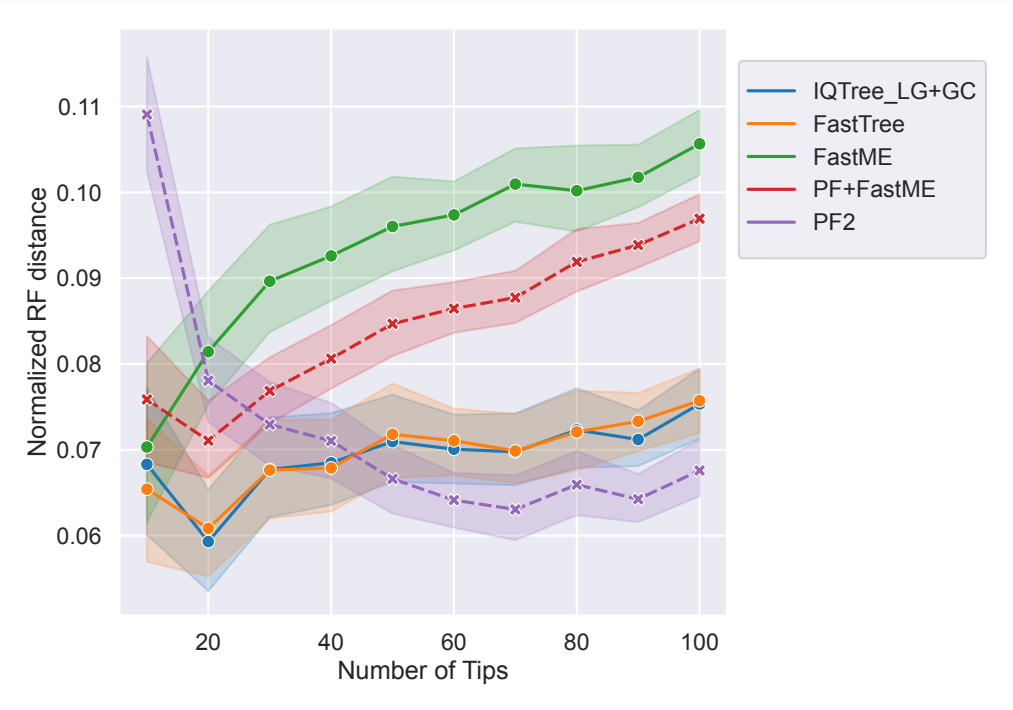
Does it work ?

Results - Training topology only



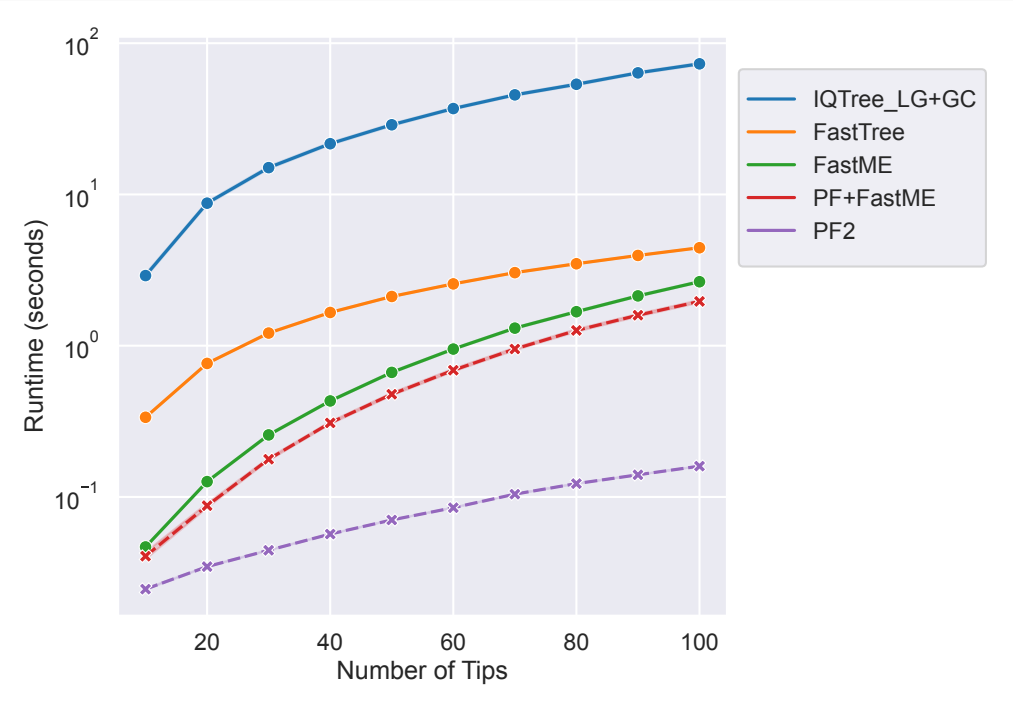
- **Length overfitting** is an **issue**

Results - Training topology only



- **Length overfitting** is an **issue**
- **Fine tuning** helps

Results - Scalability



- PF2 is **fast**
- **Faster** than PF
- ≈ 100% on **GPU**

Results - Including branch lengths

Earlier was a **pretty** picture, in fact we have **not decided** on the **parametrization**:

- **Cherry length**: Gamma, LogNormal, truncated Normal,...
- **Cherry balance**: Beta, Kumaraswamy, Triangular, truncated Normal,...
- **NJ branch lengths**: MSE is $\log PDF(\ell)$ for $\mathcal{N}(\hat{\ell}_{NJ}, 1)$

For each combination test: **bounds**, **normalizations**, **initializations**, prediction in **log scale**, etc...[1]

[1] Several hundreds of W&B runs

Results - Initial attempts with branch lengths

To test parametrizations we **overfit** PF2 on a tiny dataset: $\{(x_1, \theta_1), \ldots, (x_{50}, \theta_{50})\}$

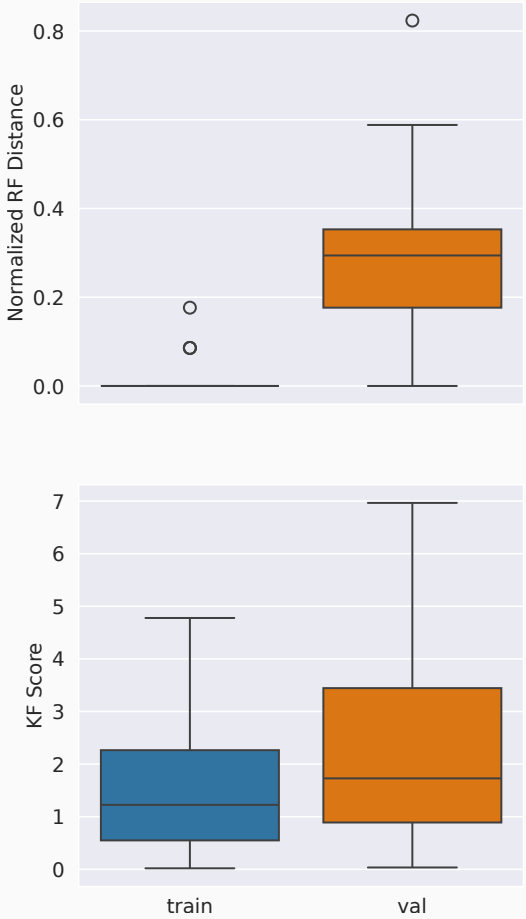
- **Cherry length**: we use a **LogNormal** and predict: $\log \mu_x$ and $\log(\mu_x/\sigma_x)$
- LogNormal **parameters** from μ_x and σ_x:

$$\mu = \log \frac{\mu_x^2}{\sqrt{\mu_x^2 + \sigma_x^2}} \quad \sigma^2 = \log\left(1 + \frac{\sigma_x^2}{\mu_x^2}\right)$$

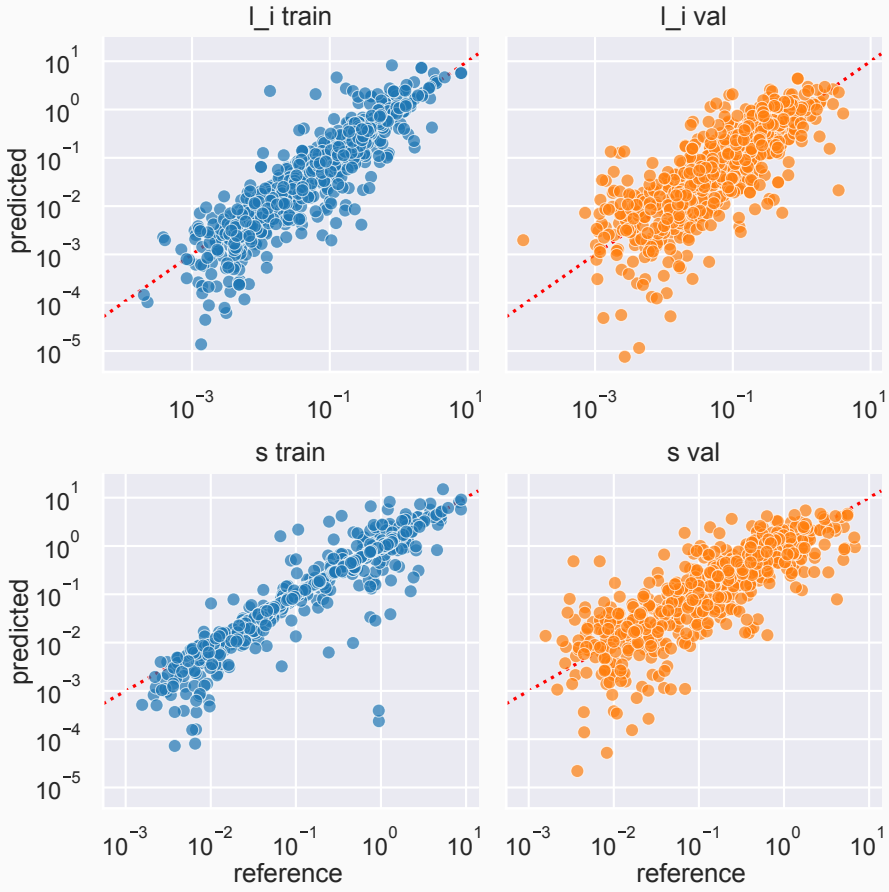
- **Predict** one **branch** ℓ_{right} instead of balance: $\mu \in [0, s]$ and $\log \sigma$ with $\sigma \in [0.1, 10]$ *(+ BatchNorm)*
- Compute **probability** with $\mathcal{N}(\mu, \sigma)$ truncated to $[0, s]$

Results - Overfitting experiment

Topology



Branch lengths



Perspectives - Where do we go now ?

Short term, **practical**:

- Branch length experiments, **settle** down on final **parametrization**
- **Train PF2** instance on a large **LG** training set

Longer term, **theoretical**:

- **Posterior** estimation: **approximation performance** of PF2?
- **Likelihood-free**: Models for which $p(\theta|x)$ is **intractable**

Perspectives - Posterior approximation study

- We can get a **"ground truth"** $p(\theta|x)$ by running e.g. RevBayes for **many iterations**[1]
- Such **distributions** have a **limited** number of **topologies**
- **Compare** topology **frequencies** between $p_{\text{RevBayes}}(\theta|x)$ and $p_{\text{PF2}}(\theta|x)$
- Check **calibration** by comparing branches of **simulated** θ and samples $p_{\text{PF2}}(\theta|x)$
- For branch lengths, maybe you know ?

Höhna et al. 2016

[1] $10^6 \approx \infty$ right?

Perspectives - Intractable likelihoods

- **Topologically** we manage to **beat** `IQTree`[1] on LG
- Can we do **better** with complex models where computing $p(\theta|x)$ is **difficult** or **intractable**?
- **Interaction** models:
  - **CherryML**, residue pair coevolution
  - **Potts** models, How do we simulate ?
  - **Epistasis** models
- Models taking **selection** into account: e.g. SelReg
- **Confident** this can **work** given our experience with **PF**

Prillo et al. 2023; Duchemin et al. 2023; Latrille et al. 2021

[1] Yay!

Conclusion

- **WIP** but we are close to truly **end-to-end likelihood-free** phylogenetic **inference**
- Still **limitations**:
  - **Better** than **PF** but **scalability** is still an **issue**
  - **Length overfitting** also an issue
- **Where do we go** once PF2 is done ?
  - Extend to **unaligned** sequence
  - Predict **Ancestral** sequences or characters
  - **Downstream** tasks: population dynamics, reconciliation, epidemiology, ecology [1]...

[1] i.e. DEELOGENY...

Thanks to:

- **Luca Nesterenko**
- **Laurent Jacob**
- **Bastien Boussau**
- **Nicolas Lartillot**
- **Philippe Veber**
- **Vicent Garot**
- **Amélie Leroy**
- **Anybody that listened to me!**

Special thanks to Jean-Zay for all the GPUs!

References

Berthet, Q. et al. (2020). ***Learning with Differentiable Pertubed Optimizers*.** In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 9508–9519.

Dong, J. et al. (2024). ***Flex Attention: A Programming Model for Generating Optimized Attention Kernels*.**

Duchemin, L. et al. (2023). ***Evaluation of methods to detect shifts in directional selection at the genome scale*.** In: *Molecular Biology and Evolution* 40.2, msac247.

Höhna, S. et al. (2016). ***RevBayes: Bayesian Phylogenetic Inference Using Graphical Models and an Interactive Model-Specification Language*.** In: *Systematic Biology* 65.4, pp. 726–736.

Jang, E. et al. (2017). ***Categorical Reparameterization with Gumbel-Softmax*.** In: arXiv:1611.01144.

Jumper, J. et al. (2021). ***Highly accurate protein structure prediction with AlphaFold*.** In: *Nature* 596.7873, pp. 583–589.

Latrille, T. et al. (2021). ***Inferring Long-Term Effective Population Size with Mutation–Selection Models*.** In: *Molecular Biology and Evolution* 38.10, pp. 4573–4587.

Lefort, V. et al. (2015). ***FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program.*** In: *Molecular biology and evolution* 32.10, pp. 2798–2800.

Nesterenko, L. et al. (2025). ***Phyloformer: Fast, Accurate, and Versatile Phylogenetic Reconstruction with Deep Neural Networks.*** In: *Molecular Biology and Evolution* 42.4, msaf051.

Prillo, S. et al. (2023). ***CherryML: scalable maximum likelihood estimation of phylogenetic models.*** In: *Nature methods* 20.8, pp. 1232–1236.

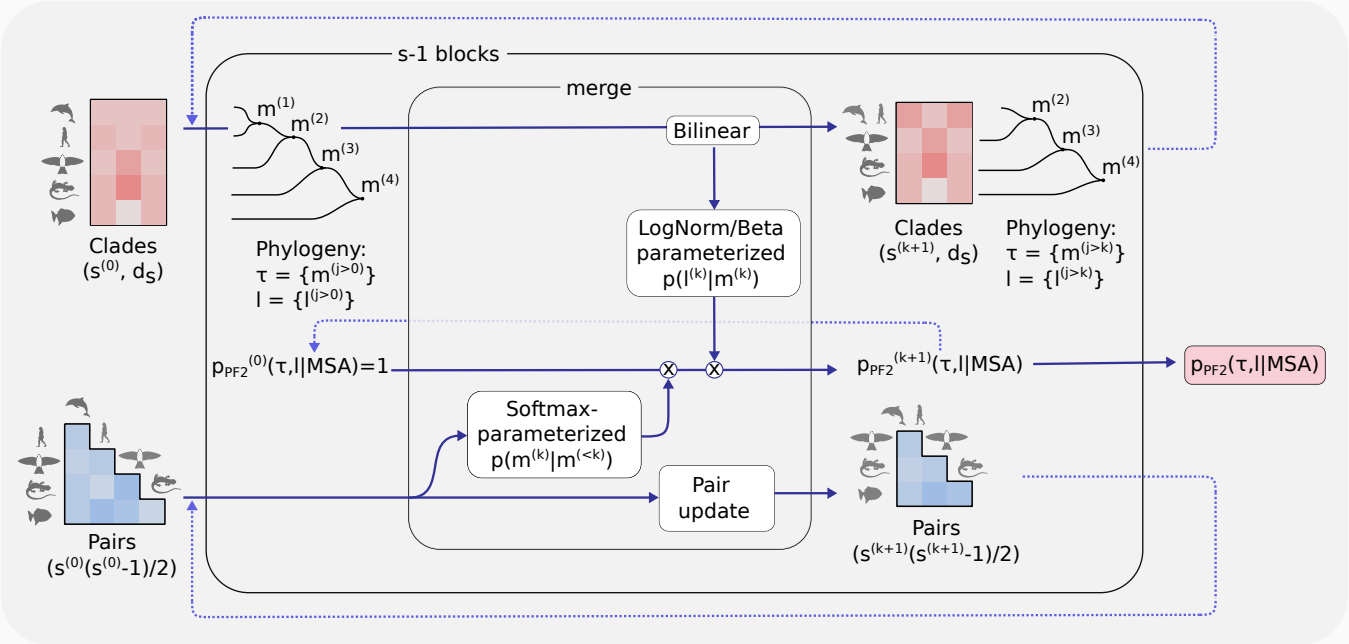
Rao, R. M. et al. (2021). ***MSA Transformer.*** In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8844–8856.

Suvorov, A. et al. (2019). ***Accurate Inference of Tree Topologies from Multiple Sequence Alignments Using Deep Learning.*** In: *Systematic Biology* 69.2, pp. 221–233.

Tang, X. et al. (2024). ***Novel symmetry-preserving neural network model for phylogenetic inference.*** In: *Bioinformatics Advances* 4.1, vbae022.

Zou, Z. et al. (2020). ***Deep residual neural networks resolve quartet molecular phylogenies.*** In: *Molecular biology and evolution* 37.5, pp. 1495–1507.

Sup. Methods - BayesNJ evaluation mode



Sup. Methods - Ensuring the merge order is unique

Ensuring a **unique order** on merges ensures that we **define a distribution**. It also keeps **training** and **sampling** comparable [1]

- On a given tree τ always **merge** the **shortest** available **cherry**
- When **sampling**, add **constraints**:
  1. Start with a $N \times N$ constraints matrix $M_{ij} = 0$
  2. At iteration k sample merge $m^{(k)} = (i, j)$ and cherry length $s^{(k)} = M_{ij} + X$
  3. **Update constraints** for cherries **available** when sampling $m^{(k)}$: $M'_{ij} = max(M_{ij}, s^{(k)}) \quad M'_{ui} = 0$
- During evaluation compute $p_{PF2}(s^{(k)} - M_{ij} | m^{(\leq k)})$

[1] Which is not the same if we use the NJ merge order

Sup. Methods - Tree simulation