

## **Origins and losses of parasitism**

an analysis of the phylogenetic tree of life with a parsimony-like algorithm

# Abstract

Anmerkung: Alle Figures richtig beschriften. Generell: Alle Figure captions sollten selbsterklärend sein. Sprich, ich sollte nur die Caption (normalerweise auch unter dem Bild) lesen und nicht den Rest des Papers und dann trotzdem verstehen können, was eigentlich passiert ist und - hilfreich - auch warum das relevant ist. (Bernhard)

# Contents

1	Introduction	4
2	Methods	7
2.1	Get data . . . . .	7
2.1.1	GloBI . . . . .	7
2.1.2	OTL . . . . .	9
2.2	Metadata analysis . . . . .	9
2.2.1	Countings . . . . .	10
2.2.2	Multifurcation . . . . .	10
2.3	Simulation . . . . .	11
2.3.1	random binary tree . . . . .	12
2.3.2	tag tree . . . . .	13
2.3.3	multifurcate tree . . . . .	14
2.3.4	maximum parsimony algorithms . . . . .	16
2.4	real data analysis . . . . .	18
2.5	Implementation . . . . .	18
3	Results	19
3.1	Influence of different parameters . . . . .	19
4	Discussion	21
	Bibliography	22

# 1 Introduction

This paper is about the further development of parsimony algorithms for non-binary trees, applied to the currently largest phylogeny synthesis tree of Open Tree Of Life, with the application to the ancestral state reconstruction of parasitism.

Anmerkung: Der erste Satz muss nochmal überarbeitet werden. Das Ziel / Ergebnis der Arbeit hat sich i

Anmerkung: Wir haben mehr einen Algorithmus getestet und an unserem spezifischen Problem angewendet, als viel selbst zu entwickeln. Allerdings haben wir den Fitch Algorithmus von binär auf multinär umgeschrieben.

Anmerkung: Mein Vorschlag einer Gliederung (jeweils ca. ein Absatz) (Bernhard)

## i) Motivation:

- Was ist das große Ziel?

Das Ziel dieser Arbeit ist die Anwendung von maximum parsimony algorithmen auf nicht binäre Bäume und auf sehr große Datensätze. Insbesondere auf das Beispiel 'Entstehung des Parasitismus' im ganzen Eukaryotischen Tree of Life.

- Was soll erreicht werden?

Wir wollen vorhandene Algorithm (Sankoff/castor) auf diese Aufgabenstellung hin testen und ihre Vorhersagekraft abschätzen. Außerdem wollen wir den Fitch algorithmus für binäre Bäume auf unser Problem erweitern und mit dem Sankoff Algorithmus vergleichen.

- Warum ist das relevant? Was könnte man dann tun?

TODO: !!!

## ii) Hintergrund:

- Was gab es in dieser Richtung bereits als ganze Ansätze oder wenn nicht, warum nicht? Woran ist es bisher gescheitert?

Bisher wurden vorallem Algorithmen für das binäre Problem entwickelt, da man wesentlich kleinere Teilbäume betrachtet hat, von welchen man auch alle Aufspaltungen kennt. Durch die Entwicklung von OTL, eines gesamten Baum des Lebens, ergibt sich das Problem, dass dieser bei weitem nicht binär ist.

Researchers of the phylogenies have been dealt with the ancestral state reconstruction in the 60s. The first methods were only brute force

TODO: Quelle, siehe Fitch: Camin

. Next came a set of parsimony algorithms such as: Fitch-parsimony [Fit71], Wagner-parsimony [SM87] ...

TODO: weitere?

With more and more data, there is now the possibility to use more information to calculate the probabilities of the ancestral states. In addition to the states of the leafs, algorithms could also use branch lengths. The likelihood based algorithms came more in interest.

Our focus came with another 'data extension'. We wanted to work with the biggest phylogenetic tree that exists at this moment, which goes over all observed species. For most

TODO: most?

species there is no phylogeny, but only a taxonomic classification.

- Welche Grundlagen sind notwendig:
  - open tree of life: Was ist das, warum relevant und überlegen als reine Ansätze?

TODO: !!!

So the biggest 'phylogenetic tree' is a synthesis of phylogenetic trees filled with a taxonomic tree given by Open Tree of Life [HSA<sup>+</sup>15]. This tree is not binary and therefore the developed algorithms are not directly applicable.

- Algorithmen: Was gibt es? Ruhig ausführlicher als hier bereits und vor allem auch nach einer Darstellung am Ende ableiten, was für uns relevant ist. Also beschreiben, wie Methode a, b, c funktionieren und dann abwägen, was daher für Dich am relevantesten ist.

TODO: !!!

Anmerkung: GloBI und OTL in der Einleitung vorstellen. (Emanuel)

### iii) Outlook/Structure of this work

In this work, we have looked at the algorithms that are generally suited to our data, to develop them further for the not binary case, and finally to compare their usability with our sythesis tree.

We have decided to consider only parsimony algorithms since we have no information on branch lengths and no other additional information like different transition probabilities of our states.

## 2 Methods

Anmerkung: Graphischen Workflow hinzufügen, mehr Motivation und genauer. (Bernhard)

This work consist of two aspects. One is the application of the algorithm to our question the other a simulation of this to proof the credibility of our methods.

The resulting procedure is as follows:

- i) Get the real tree and real data for the leaf nodes.
- ii) Get metadata of these for a realisitc the simulation.
- iii) Build and run the simulation.
- iv) Evaluation of parameters for the simulation and the real problem.
- v) Run the resulted algorithm on the original data.
- vi) Evalute and interpret results. -> Origins etc...

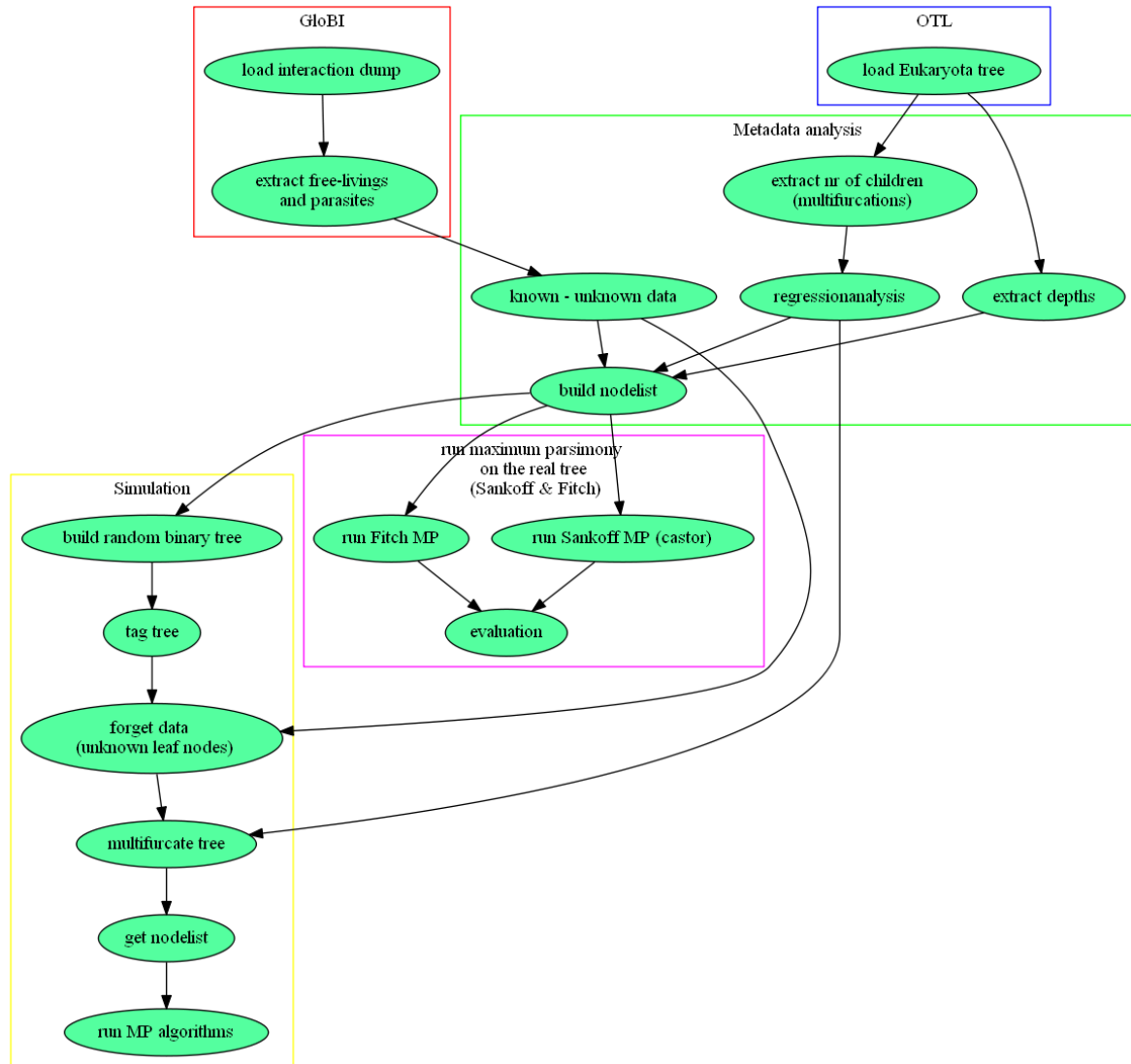
### 2.1 Get data

#### 2.1.1 GloBI

For tagging our leave nodes, we use the GloBI (Global Biotic Interactions) database. This database consists of entries of the form: species A interacts with B.

We appointed some interactions, where we know from the biological perspective that the species source or target has to be a parasite or a free-living species. These are the following:

**Figure 2.1: Workflow**




---

```

freeliving_source = ["parasiteOf", "pathogenOf"]
freeliving_target = ["hasParasite", "hasPathogen"]
parasite_source = ["preysOn", "eats", "flowersVisitedBy", "hasPat
parasite_target = ["preyedUponBy", "parasiteOf", "visitsFlowersOf

```

---

We build two lists: parasites and free-livings, and add the source or targets of an interaction to these.



TODO: klar? Oder Beispiel bringen? (Katze isst Maus -> Katze ist Freilebend)

TODO: ott id hier erwähnen?

TODO: einige spezies nicht mit einbezogen, da sie keine OTT id haben, hier könnte man noch verbessern

TODO: You can find all interaction types here: <https://github.com/jhpoelen/eol-globi-data/blob/master>

## 2.1.2 OTL

TODO: Open Tree of Life

## 2.2 Metadata analysis

Properties of real Data - Metadata analysis

There are some Parameters to find out or notice:

Anmerkung: hier unterscheiden zwischen einfachen Zahlen (countings / one dimensional) und komplexen variablen (Emanuel)

Anmerkung: arg außerdem zu unterscheiden: Biologische Parameter (transition probabilities), fixe Daten und zu berechnende (multifurcation,...) (Emanuel)

- transition probabilities of tags
- multifurcation
- nr of parasites to free-living
- nr of unknown nodes

Anmerkung: multifurcations und unknown leave nodes sind ähnlich beeinflusst (Emanuel)

TODO: Hier darauf hinweisen, dass wir für multifurcations und unknown data den einfluss auf die Algorithmen

## 2.2.1 Countings

Some simple countings...

nr of parasites, free-living and unknown nodes ... from whole tree, subtrees...

Name	Number of
interactions	5346414
freeliving species	51337 (distinct)
parasite species	47332 (distinct)
unused possible species	57352 (source), 809993 (target)
Eukaryota	241974 internal nodes, 2293463 leaf nodes

$$\frac{100}{2293463 * (51337 + 47332)} \approx 4.3 \rightarrow \text{only 4.3\% of leaf nodes are filled with information}$$

## 2.2.2 Multifurcation

Anmerkung: Description/Definition of Multifurcation -> to the Intro (Bernhard)

Anmerkung: Regressionsanalyse (Bernhard)

Anmerkung: poisson glm: #children ~ I phylum -> R fitdist p (Emanuel)

Anmerkung: different parameters for different taxa? (Emanuel)

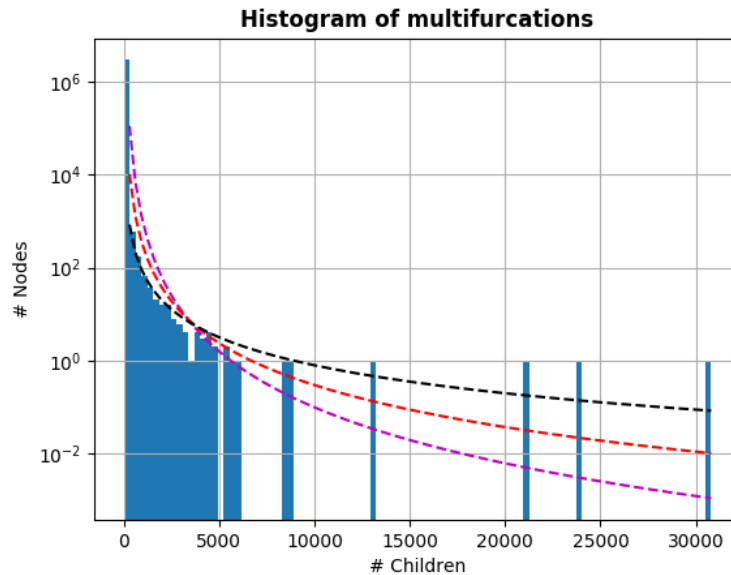
One property of the tree is its ridge of multifurcation. For this we collected for every node its number of children (degree -1), and plottet this in an histogram, how much nodes we have for every degree. Then we tried to find a function which best discribes this behavior. In figure 2.2 is this plot shown with three functions, which came **TODO: close(st)** to this goal:  $\frac{1000000000000000}{x^4}$  (magenta),  $\frac{300000000000}{x^3}$  (red) and  $\frac{80000000}{x^2}$  (black).

From the biology we know, that this multifurcation is very clouded. Some parts of the tree are well known and others have not been very interesting for the research so far. So we also wanted to know how the multifurcations behave in some interesting subtrees.

Anmerkung: do we? Is that really universal knowledge or could it be supported by data or a citation? (B

**TODO: subtrees... there is some old stuff...**

**Figure 2.2:** Histogram of multifurcations and fitting functions



TODO: other subsections here...

## 2.3 Simulation

Anmerkung: Motivate the goal of the simulation (Bernhard)

- build random binary trees, tag these (parameters: parasites vs free-living, beta-distribution)
- run fitch-parsimony, wagner-parsimony, our parsimony like algorithm
- build not binary tree (poisson distribution?)
- run new algorithms
- compare trees (distances)

i) build random binary trees

ii) tag tree

- iii) multifurcate tree
- iv) run maximum parsimony algorithms
  - Fitch
  - Sankoff (Castor package)
  - my algorithm
- v) Evaluation

### 2.3.1 random binary tree

Anmerkung: Again, motivate first, why this is required and why you choose this solution (Bernhard)

To get a random binary tree, I used the Phylo package from biopython. They offer a randomized function which returns a BaseTree <sup>1</sup>:

---

```
from Bio import Phylo
Phylo.BaseTree.Tree.randomized(number_leaf nodes)
```

---

From the BaseTree class: Anmerkung: trivial, does not give real info (Emanuel)

---

```
def randomized(cls, taxa, branch_length=1.0,
               branch_stddev=None):
    """Create a randomized bifurcating tree given a list
       of taxa.
       :param taxa: Either an integer specifying the number
                     of taxa to create (automatically named taxon#),
                     or an iterable of taxon names, as strings.
       :returns: a tree of the same type as this class.
    """
```

---

TODO: Zitat von BaseTree und buildTree.py

---

<sup>1</sup><https://github.com/biopython/biopython/blob/master/Bio/Phylo/BaseTree.py>

### 2.3.2 tag tree

Anmerkung: instead of 'tag tee' simulating stages and transitions between them (Emanuel)

At this point we want one fully tagged tree, and one less tagged tree which looks like our real data.

Let's say the first specie (the root node) was free-living (start with a parasite without a host makes no sence). For every transition from a node to his child, we take a random number from the father distribution. We decided that from the biological perspective a beta distribution reflects our transition probabilities best (see Figure 2.1 TODO: ref einfügen ).

Anmerkung: why is that the case and why is that from a biological perspective? (Bernhard)

Anmerkung: I'd rather say that to ensure that the parameter of the binomial distribution is restricted to the  $[0,1]$  interval, we model it... (Bernhard)

For example when our father node was free-living, then we take from the free-living beta distribution. Is the number under the threshold for beeing parasite, we get a change and tag the current node as parasite. Otherwise we tag it as free-living.

With this procedure we traverse through the tree from the root to every leaf node. A part of this code you see here:

---

```
from numpy import random
if father_tag == 0:
    # freelifving_distribution:
    new_random = random.beta(a=A_FL, b=B_FL)
else:
    # parasite_distribution:
    new_random = random.beta(a=A_P, b=B_P)
tag = 0          # -> FL
if new_random < percentage_parasites:
    tag = 1      # -> P
```

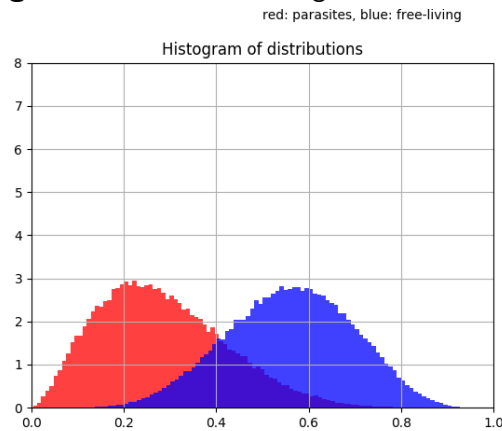
---

TODO: Bessere Beschriftung, Plot neu erstellen! U.a. mit threshold

We save each tag with the associated node ID in a nodelist.

Anmerkung: simulationg loss of information (1) states (2) topology/multifurcation (Emanuel)

**Figure 2.3: 60% Free-living - 40% Parasites**



The real tree has much less information, we have only information from some current species (leaf nodes) and **TODO: and probably negligible internal nodes** .

To simulate our real tree we save for every node an empty placeholder except for some leaf nodes. There we save the tags again. The amount of these unknown information is one parameter, which we got from our real tree. Or which we can change to **TODO: ...**

**TODO: Was hiervon gehört ind Methoden, was schon in Implementierung oder ganz woanders hin?**

### 2.3.3 multifurcate tree

**Anmerkung: simulating loss of information (Emanuel)**

Another parameter is the nature and strength of the multifunction of the tree, since we do not have a binary tree in the real case. After several measurements and analyzes, which we explain in **TODO: section/chapter x** , **Anmerkung: fit, justification (Emanuel)** we decided to use a  $\frac{1}{x}$  distribution, where  $x$  is the depth of a node. This means, how deeper we are, how less information we have.

We traverse through the tree and pick a random number between 0 and 1. If random number is smaller as our limit ( $\frac{1}{x}$ ), than we forget the node and hang every child to the father node of the current node.

Anmerkung: poisson process -> fit that distribution, include depth as a predictor, see if significant (Eman)

---

```
from numpy import random
from utilities import Helpers

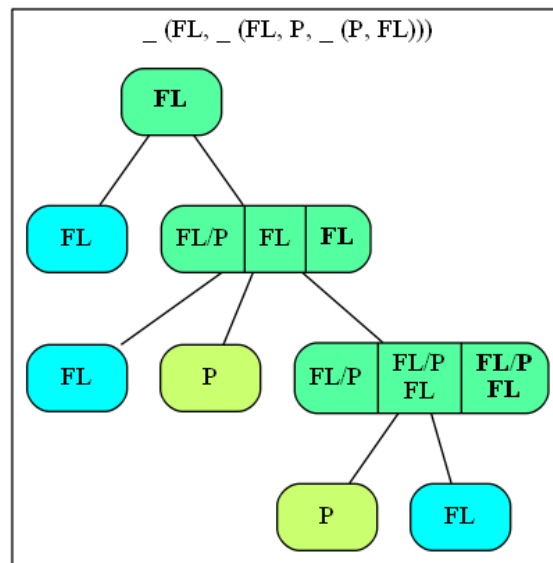
def get_non_binary_tree(subtree, nodelist):
    i = 0
    while i != len(subtree.clades):
        if subtree.clades[i].is_terminal():           # is leaf node
            i += 1
        else:
            element = Helpers.find_element_in_nodelist(subtree.clades[i])
            limit = get_limit(element[1])
            new_random = random.uniform()             # choose if v
            if new_random < limit:                     # or new_rand
                subtree.clades += subtree.clades[i].clades # add children
                del subtree.clades[i]                  # delete inte
            else:                                       # if we don't
                get_non_binary_tree(subtree.clades[i], nodelist)
# otherwise the children are in the current clade array
            i += 1
    return

def get_limit(depth):
    limit = 1 - 1 / ((depth + 3) / 4)
    if limit < 0.1:
        limit = 0.1
    return limit
```

---

Wir lassen das Limit nicht beliebig klein werden, sondern beschränken es auf 0.1.

Figure 2.4: bla



## 2.3.4 maximum parsimony algorithms

### Fitch maximum parsimony

Described from [COO98] + others ... - implemented for multifurcating trees

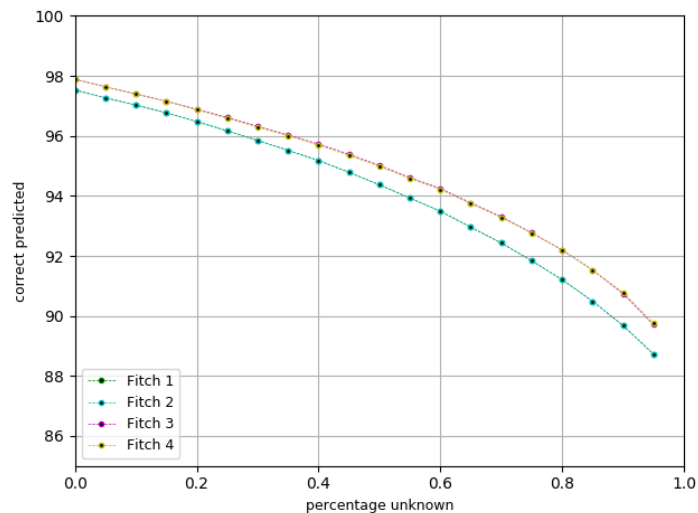
Fitch algorithm for binary trees:

Der Baum hat die folgende Struktur: Alle inneren Knoten sind leer. In den Blattknoten befindet sich entweder das Tag FL oder P, oder deren Vereinigung, wenn es sich um einen unknown node handelt.

Der Fitch Algorithmus ist aufgeteilt in drei Schritte, in welchen man jeweils durch den Baum traversiert. Schritt 1 beginnt von de Blättern aus, da sich dort zu Beginn die einzige Information befindet. Für jeden Knoten gilt, wenn seine Kinder schon Information enthalten, dann bilde die Schnittmenge der Tags und schreibe diese als Information in den aktuellen Knoten. Ist die Schnittmenge leer, dann schreibe die Vereinigung aller möglichen Tags in den Knoten. Für alle Kinder, die noch keine Information haben, führe diesen Schritt erst für diese aus. Schritt zwei geht von den Kindern der Wurzel bis zu den Vätern der Blätter. Jeder Knoten bekommt einen zweiten Tag, der sich aus der Vereinigung des Tags des Vaterknoten



Figure 2.5: bla



und der Geschwisterknoten zusammensetzt. Ist diese leer, bekommt der Knoten wieder die Vereinigung aller Tags, also  $\{FL, P\}$  als Tag.

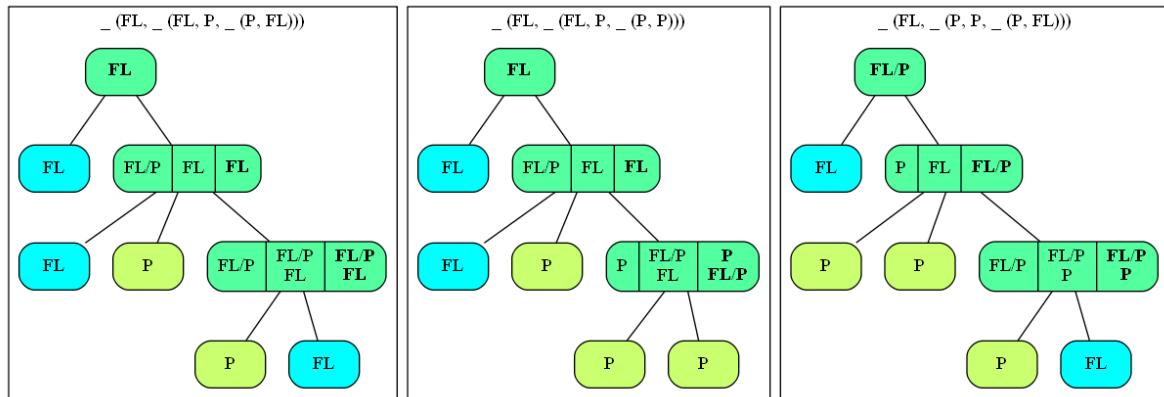
Hier gibt es einige Möglichkeiten, wie dieser Schritt genau aussieht. 1. Version: Es wird nur der erste Tag vom Vaterknoten genutzt. Außerdem wird von den Geschwisterknoten zuerst der Schnitt gebildet, und danach vom Ergebnis nochmal mit dem Vaterknoten zusammen. (Immer wenn der Schnitt leer ist, ist das Ergebnis die Vereinigung aller Tags, also  $\{FL, P\}$ . Auch im folgenden...) 2. Version: Es wird nur der erste Tag vom Vaterknoten genutzt. Er wird zusammen mit den Geschwistertags genommen und direkt ein Schnitt aller Mengen gebildet. 3. Version: Es werden alle vorherigen Tags vom Vater genutzt und von diesen ein Schnitt gebildet. Das selbe gilt für die Geschwistertags. Und dann wird ein dritter Schnitt zwischen den Ergebnissen gebildet. 4. Version: Es werden alle Tags genutzt und direkt in einem Schnitt zusammengekommen.

Der Finale Schritt traversiert nochmal über den Baum und bildet aus den zwei Tags pro Knoten einen finalen Tag, indem wieder der Schnitt der beiden Tags das Ergebnis ist.

Ich habe diese Versionen mit 100 Bäumen mit 10000 Blattknoten und der Verteilung 60% FL zu 40% P simuliert. Bei 90 % unbekannten Knoten lag Version 1 zu 89.67 %, Version 2 zu 89.67 %, Version 3 zu 90.72 % und Version 4 zu 90.74 % richtig.

How to extend Fitch for multifunction?:

**Figure 2.6: bla**



## Sankoff

Maximum parsimony algorithm from Sankoff implemented in the R package castor.

## my Algorithm

## 2.4 real data analysis

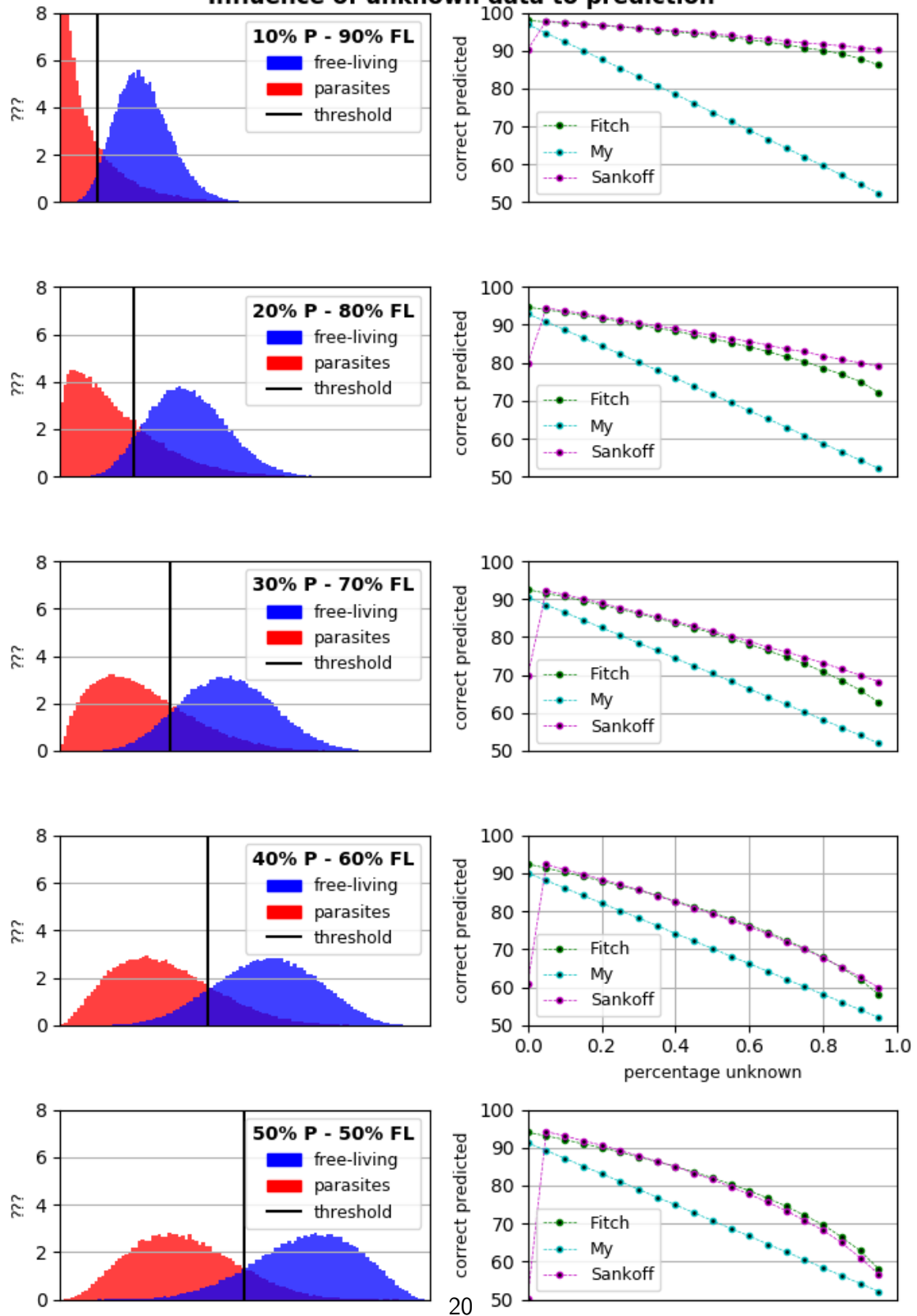
- Import tree
- Import interactions
- run castor algorithm / and others?
- interpret results (leave one out)

## 2.5 Implementation

## 3 Results

### 3.1 Influence of different parameters

**Figure 3.1: Influence of unknown data to prediction**  
**Influence of unknown data to prediction**



## 4 Discussion

Wie gut ist der randomisiert erstellte Baum?

Wie gut kommt unsere Simulation an die echte Datenlage heran.

Fehlerquote der Daten an sich?

Wie gut ist unsere Datenlage? 3 mio Knoten, 1.8 named species (leaf nodes), 200.000 leaf nodes mit Information.

Simulation von subtrees

Welche Teile des Baumes sind gut, an welchen muss noch viel geforscht werden.

Wieviele Origins haben wir gefunden, was bedeutet diese Zahl?

Parameter der Simulation:

- Wie ist die Verteilung der vergessenen internen Knoten? Zum Wurzelknoten hin mehr vergessen?
- Wie sehen die Übergangswahrscheinlichkeiten aus von P->FL und andersherum?
- Verteilung Parasiten zu Freilebend zu keine Information

Selecting of the 'right' / best Distribution

# Bibliography

- [Fit71] FITCH, Walter M.: Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology. In: *Systematic Biology* 20 (1971), Nr. 4, 406-416. <http://dx.doi.org/10.1093/sysbio/20.4.406>. – DOI 10.1093/sysbio/20.4.406
- [HSA<sup>+</sup>15] HINCHLIFF, Cody E. ; SMITH, Stephen A. ; ALLMAN, James F. ; BURLEIGH, J. G. ; CHAUDHARY, Ruchi ; COGHILL, Lyndon M. ; CRANDALL, Keith A. ; DENG, Jiabin ; DREW, Bryan T. ; GAZIS, Romina ; GUDE, Karl ; HIBBETT, David S. ; KATZ, Laura A. ; LAUGHINGHOUSE, H. D. ; MCTAVISH, Emily J. ; MIDFORD, Peter E. ; OWEN, Christopher L. ; REE, Richard H. ; REES, Jonathan A. ; SOLTIS, Douglas E. ; WILLIAMS, Tiffani ; CRANSTON, Karen A.: Synthesis of phylogeny and taxonomy into a comprehensive tree of life. In: *Proceedings of the National Academy of Sciences* 112 (2015), Nr. 41, 12764-12769. <http://dx.doi.org/10.1073/pnas.1423041112>. – DOI 10.1073/pnas.1423041112
- [SM87] SWOFFORD, David L. ; MADDISON, Wayne P.: Reconstructing ancestral character states under Wagner parsimony. In: *Mathematical Biosciences* 87 (1987), Nr. 2, 199 - 229. [http://dx.doi.org/https://doi.org/10.1016/0025-5564\(87\)90074-5](http://dx.doi.org/https://doi.org/10.1016/0025-5564(87)90074-5). – DOI [https://doi.org/10.1016/0025-5564\(87\)90074-5](https://doi.org/10.1016/0025-5564(87)90074-5). – ISSN 0025-5564