Benchmarking the OptiFit Algorithm

Brodie Mumphrey & Kelly Sovacool

What are OTUs and why do we use them?

Operational taxonomic unit

- Groups of organisms
  - Groups can be based on whatever metric makes sense for a given study.
- Due to the difficulty in phenotyping bacteria, bacterial OTUs are often generated based on genetic distance.
- Genetic distance is most often approximated by 16s rRNA hypervariable regions (we use the V4 region).
- Bacteria are considered to be the same “species” at 97% sequence similarity.
- Assigning high-quality OTUs is important for downstream analyses of the microbial community.

OTU clustering

Previous OTU clustering algorithms

Closed reference/phylotyping

Fits samples to a reference database (e.g. silva, greengenes).

Pros

- Fast.
- Taxonomic classification when a sequence fits.
- In a world with a perfect database, this method would give perfect classifications.

Cons

- Reference databases are biased towards easily cultured bacteria.
- Different reference databases may conflict with each other.

Previous OTU clustering algorithms

De novo clustering

Clusters sample sequences without a reference database.

Pros

- Agnostic to outside data.
- Doesn't throw away never-before-seen bacteria.

Cons

- Agnostic to outside data.
- Computationally expensive.
- Based on "arbitrary" cutoffs, such as 97% similarity.
- Based on not-completely-true assumptions such as constant rate evolution of the 16s rRNA V4 region.
- Hard to know how "good" a final set of OTUs is.

Previous OTU clustering algorithms

Open reference clustering

Fits samples to a reference database, then performs de novo clustering on whatever did not fit.

Pros

- Gains information from well curated databases.
- Keeps information from previously uncharacterized bacteria.

Cons

- Derives two sets of OTUs based on different metrics.

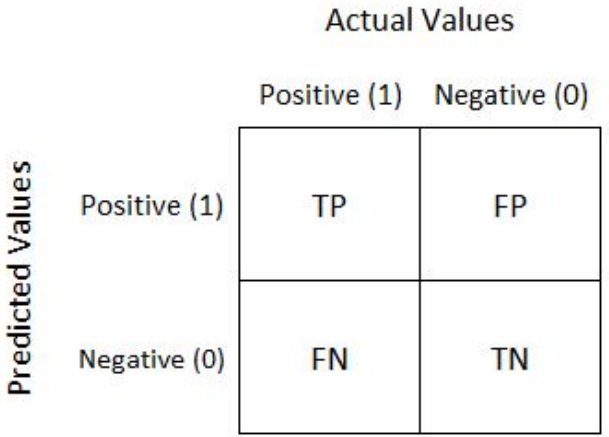
OptiClust, an Improved Method for Assigning Amplicon-Based Sequence Data to Operational Taxonomic Units

Sarah L. Westcott, Patrick D. Schloss

Department of Microbiology and Immunology, University of Michigan, Ann Arbor, Michigan, USA

Mothur clustering algorithm: OptiClust

- De novo clustering method.
- Moves from continuous distance to a binary decision of similarity.
  - Similar if >97% sequence similarity, otherwise not similar.
- Drastically reduces compute time.
- Uses the confusion matrix to measure the quality of OTU assignments.



Mothur clustering algorithm: OptiClust

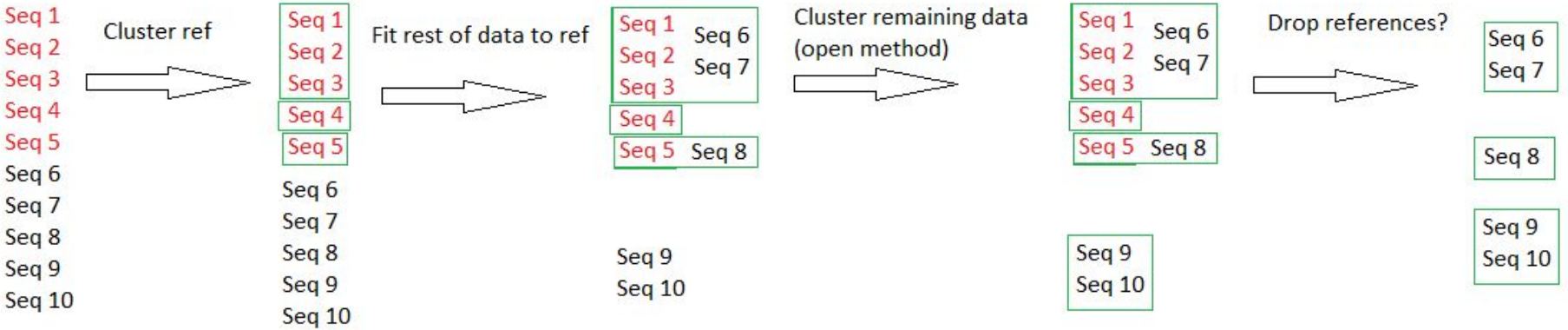
- Algorithm iteratively assigns samples to OTUs by maximizing the MCC.
- Matthews Correlation Coefficient:

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- OptiClust Performance:
  - Time & memory complexity: ~n^2
  - 94 times faster than VSEARCH average neighbor.
  - Just as fast as VSEARCH distance-based greedy clustering.
  - Better MCC values than VSEARCH by ~15%.

OptiFit: an extension of OptiClust

- Uses the Matthews Correlation Coefficient in the context of a reference fitting algorithm.
- Fits samples to a clustered reference.



OptiFit Analysis

- Variables of interest:
  - Clustering method (open vs closed).
  - Inclusion of reference in the final output.
  - Effectiveness on samples from different sources (human, marine, murine, soil).

OptiFit Analysis

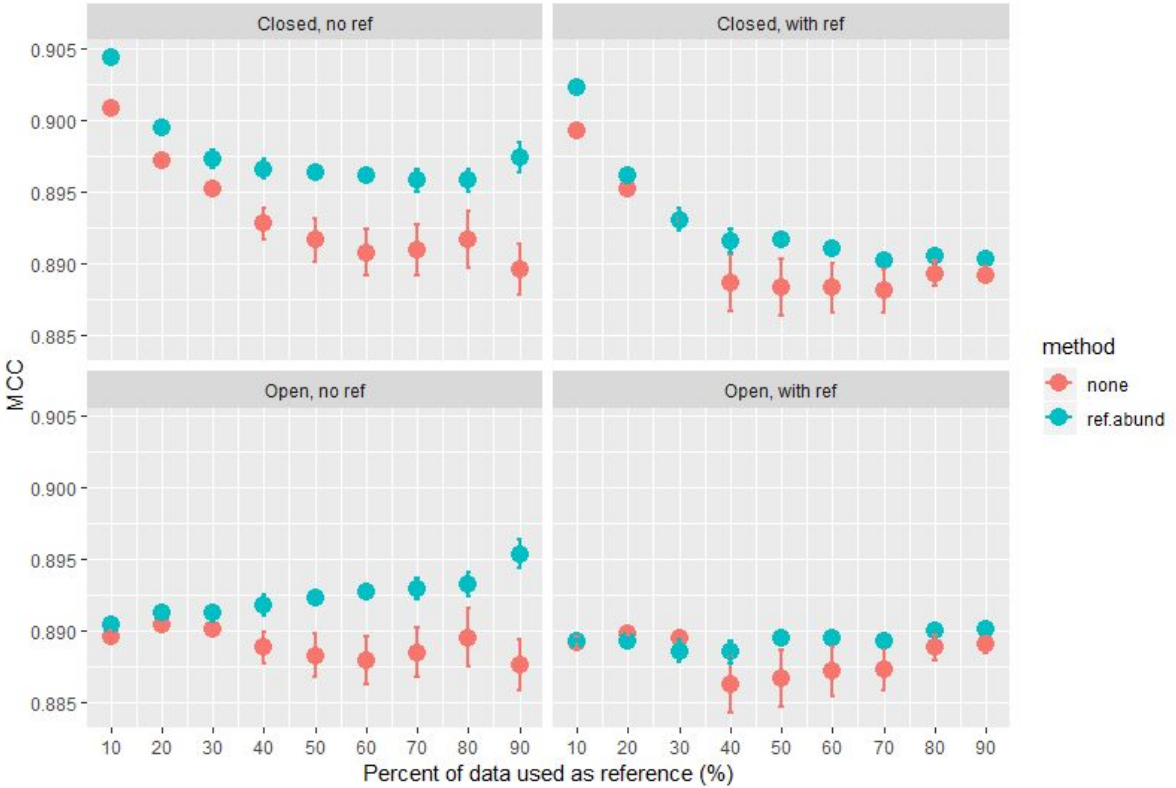
Dataset as its own reference

	50% reference	20% reference	80% reference
Seq1	Seq1	Seq1	Seq1
Seq2	Seq2	Seq2	Seq2
Seq3	Seq3	Seq3	Seq3
Seq4	Seq4	Seq4	Seq4
Seq5	Seq5	Seq5	Seq5
Seq6	Seq6	Seq6	Seq6
Seq7	Seq7	Seq7	Seq7
Seq8	Seq8	Seq8	Seq8
Seq9	Seq9	Seq9	Seq9
Seq10	Seq10	Seq10	Seq10

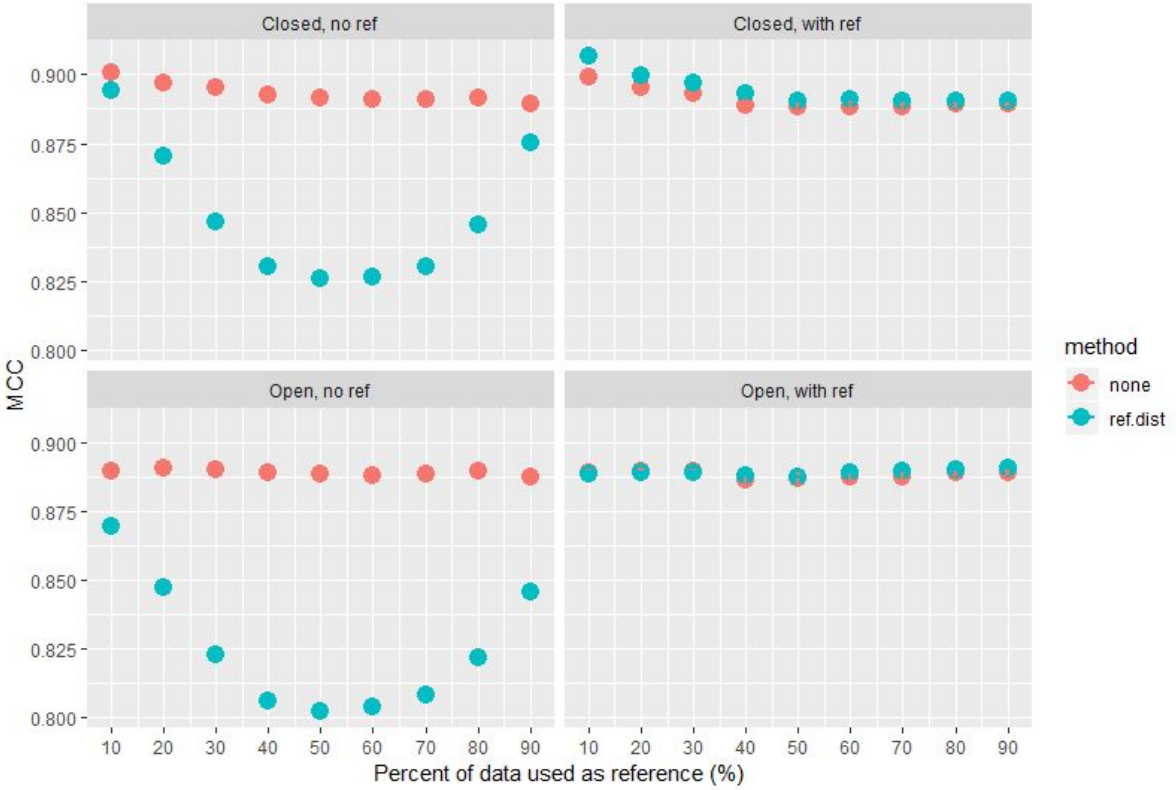
OptiFit Analysis

- Variables of interest:
  - Clustering method (open vs closed).
  - Inclusion of reference in the final output.
  - Effectiveness on samples from different sources (human, marine, murine, soil).
- Additional variables when using dataset as its own reference:
  - Percent of data used as the reference.
  - Reference selection method:
    - sample distances, sample abundance, random.

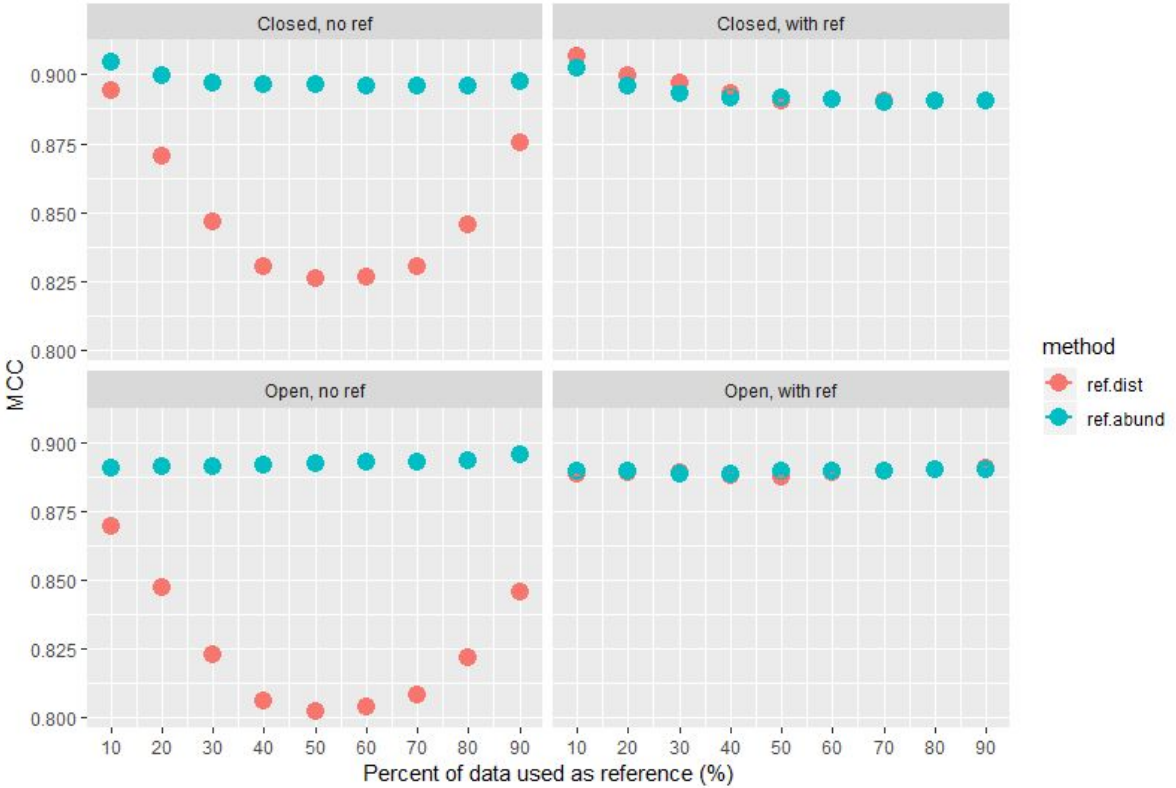
What is the best way to select samples for the reference?



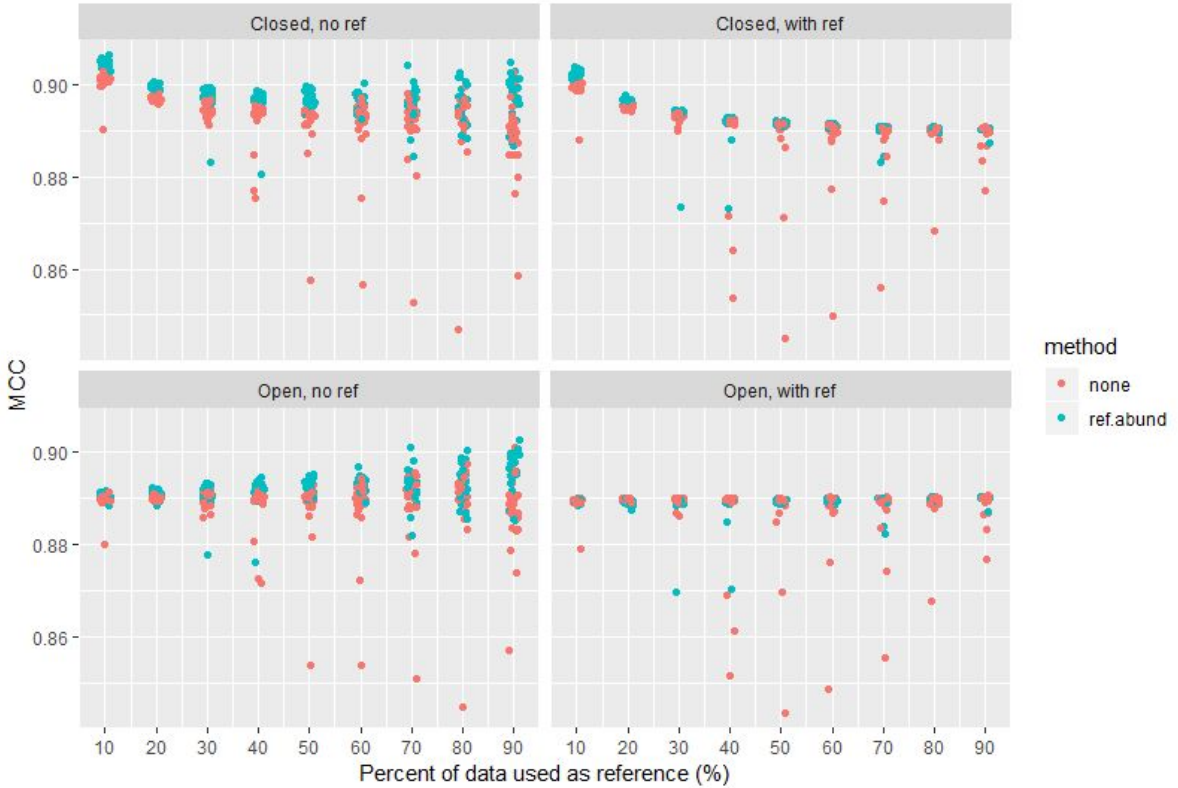
What is the best way to select samples for the reference?



What is the best way to select samples for the reference?

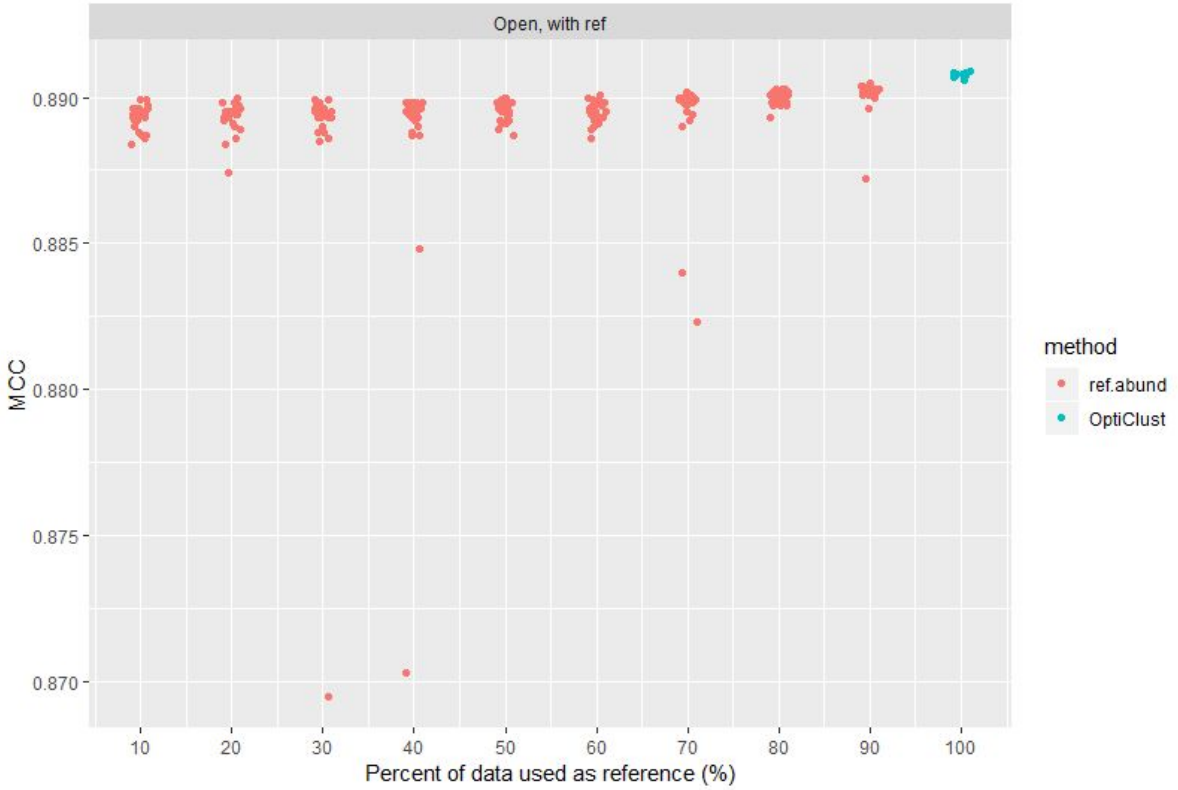


How sensitive is OptiFit to random variation?



How much of the dataset is needed to generate a good reference?

How do OptiFit MCC values compare to OptiClust?



OptiFit Characterization - How fast is OptiFit?

- Hit some roadblocks preventing us from collecting concrete data.
  - The actual clustering part of OptiFit is a subroutine of the whole function.
- Rough benchmarks manually taken with a stopwatch.
  - De novo clustering
    - OptiClust: ~85 seconds
    - OptiFit: ~90 seconds
  - Fitting to a clustered reference
    - OptiClust: ~85 seconds
    - OptiFit: ~8 seconds

Takeaways

- Selection of samples for a reference should be weighted by abundance.
- When selecting reference by abundance, OptiFit is robust against random variation.
- OptiFit produces MCC values that are on par with OptiClust when run in similar scenarios.
- OptiFit is significantly faster than OptiClust when fitting a sample to a relatively large reference database.
- For De Novo clustering, stick to OptiFit.

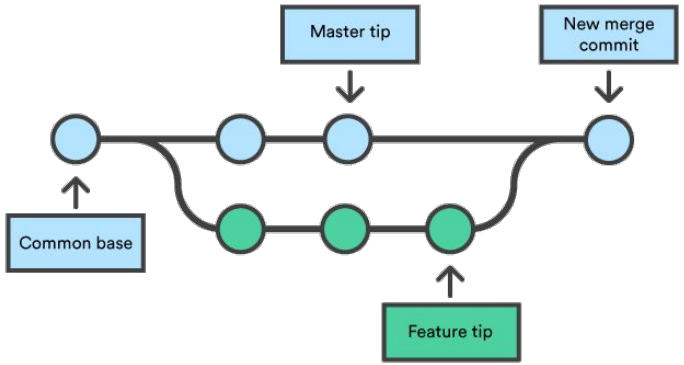
Unanswered questions

- How much of the dataset is needed to generate a good reference?
- How does OptiFit perform when fitting to large outside databases like Silva and Green Genes?
- How does OptiFit perform vs other fitting algorithms? (e.g. VSEARCH)
- How does OptiFit perform on data from different sources?

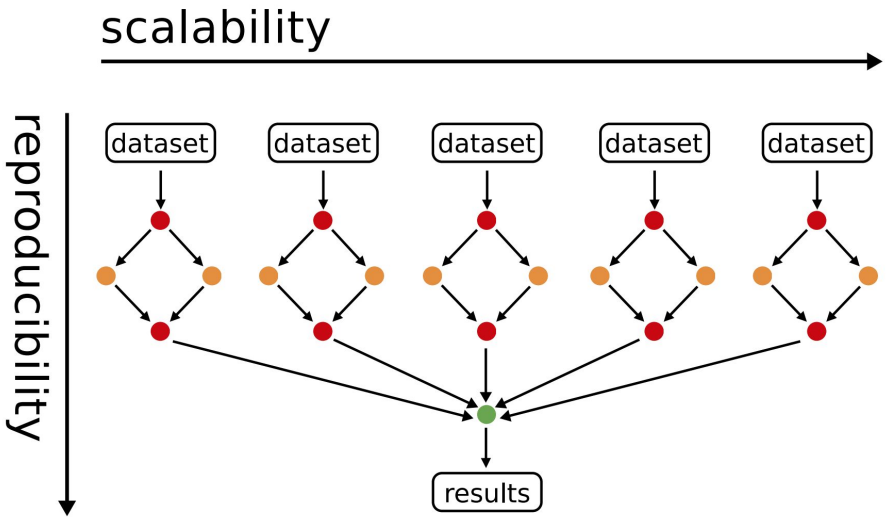
OptiFit Analysis - Implementation

- Needs to be reproducible, reusable, scalable.
- Solution: version control + a workflow management system.

Git branching



Workflow Management



Why not just run mothur manually?

The full analysis pipeline for one dataset:

```
Job counts:
        count   jobs
        1       aggregate_sensspec
        1       all
        1       calc_seq_dists
        7200    cluster
        3600    copy_cluster
        14400   copy_fit_input
        14400   fit
        1       plot_sensspec
        360     prep_reference_from_dataset
        360     prep_weighted_subsample
        360     split_weighted_subsample
        40684
```

Snakemake: Python-based workflows

- Designed by & for bioinformatics.
- Inspired by GNU make.
- Describe rules by input & output files.
- Automatically determines dependencies between the rules.
- Automatically creates missing output directories before executing jobs.
- Can execute independent rules in parallel.
- Run Bash & Python directly in rules.
- Call R & Python scripts from rules.
- Execute in a HPC environment.
- Easily measure runtime & memory usage of rules.

Python code everywhere

- Include as much or as little Python code as you want in the Snakefile.
- Access variables, functions, classes, etc. from within the run directive.

```
def do_something(args):
    # insert python code here

rule process_fasta:
    input:
        “data/{dataset}.fna”
    output:
        “results/{dataset}.txt”
    run:
        result = do_something(input)
        # insert python code here
```

Call external scripts

Snakemake rule

```
rule plot_results:
    input:
        "path/to/inputfile",
        "path/to/other/inputfile"
    output:
        "path/to/outputfile",
        "path/to/another/outputfile"
    script:
        "path/to/script.R"
```

R script

```
infilename <- snakemake@input[[1]]

outfilename <- snakemake@output[[1]]

# insert more R code to do something
```

Execute shell commands from rules

```
rule calc_seq_dists:
    input:
        'data/{dataset}.fasta'
    output:
        'data/{dataset}.dist'
    params:
        mothur=config['mothur_bin_path'],
        outdir='results/{dataset}/'
    benchmark:
        'benchmarks/{dataset}/calc_seq_dists.log'
    log:
        'logfiles/{dataset}/calc_seq_dists.log'
    shell:
        '{params.mothur} "#set.logfile(name={log});
            set.dir(output={params.outdir});
            dist.seqs(fasta={input[0]}, cutoff=0.03)"'
```

Challenges

- Mothur assumes only one instance is running on any set of input files.
  - Running multiple mothur jobs in parallel on the same input files causes clashing between temporary files, unless you copy input to separate directories.
- Snakemake features for running on a cluster are new(ish) and not well-documented.

What I learned - Brodie

So many languages

- R tidyverse functions
- Python/Snakemake
- Bash
- Git
- Flux/Torque/PBS
- The “language” of microbiome analysis and mothur

All of the different ways that OTUs can be created

The usefulness of the confusion matrix

How quickly small inefficiencies add up

What I learned - Kelly

Skills:

- Mothur
- R tidyverse
- Advanced Snakemake features
- Collaborating with git

Knowledge:

- General microbiome research techniques