q2-sidle

Release 2020.8

Jan 06, 2021

Contents:

1 Installation **3**

2 Database Preparation **5**
- 2.1 Filtering the Database . . . 6
- 2.2 Prepare a regional database . . . 7
- 2.3 TL;DR: Database Preparation . . . 8

3 Read Preparation **11**
- 3.1 Demultiplex your reads . . . 12
- 3.2 Denoise reads with your favorite algorithm . . . 13
- 3.3 Check your tables . . . 14
- 3.4 TL;DR: Read Preparation . . . 14

4 Sequence Reconstruction **17**
- 4.1 Regional alignment . . . 17
- 4.2 Table Reconstruction . . . 18
- 4.3 Taxonomic Reconstruction . . . 20
- 4.4 Reconstructing the Phylogenetic Tree . . . 21
- 4.5 Next Steps: Analysis! . . . 22
- 4.6 TL;DR Reconstruction . . . 22

5 Parallel Processing in Sidle **25**

6 Indices and tables **27**

SMURF Implementation Done to acceLerate Effeciency

Sidle is a python version of the Short MUliple Reads Framework (SMURF) algorithm originally developed by Fuks et al (2018) with a novel tree-building solution. This allows the reconstruction of multiple short, fragmented amplicons against a known database to improve the resolution fo the reconstructed community over single amplicons. It can also be used for meta-analysis to provide improved asv-based resolution.

CHAPTER 1

Installation

The current working version is a qiime2-based implementation. It requires that you have installed qiime2 according to the installation instructions and have activated the environment. In addition to vanilla qiime2, you will also need to install RESCRIPt; these installation instructions will guide you through that process.

..note:: Sidle has been tested against qiime2-2020.11 and later; it may not function with earlier versions of qiime2.

One you have activated yoru qiime2 enviroment, sidle can be installed with the following commands.

```
conda install dask regex
conda install -c conda-forge -c bioconda -c qiime2 -c defaults xmltodict
pip install git+https://github.com/bokulich-lab/RESCRIPt.git
pip install git+https://github.com/jwdebelius/q2-sidle
qiime dev refresh-cache
```

You can test the installation by running

```
qiime
```

You should see a print out which includes `sidle`.

Now, you're all ready to start using Sidle.

CHAPTER 2

Database Preparation

If you do not already have a database for reconstruction, you will need to prepare your own. (We recommend checking the Resources page to see if you can find one that suits your needs already.) Database preparation will only need to be done once for each database, primer pair and kmer length, since the reference files can be reused.

You will need:

- A working sidle *installation*
- A list of the forward and reverse files used to amplify each region of interest
- A reference database of your choice
- Patience to run the extraction

You can start the tutorial by downloading the database sequences and taxonomy. These have already been imported into qiime2 and as Artifacts.

```
mkdir -p sidle_tutorial
cd sidle_tutorial
wget https://github.com/jwdebelius/q2-sidle/raw/main/docs/tutorial_data/database.zip
unzip database.zip
cd database
```

After you run this command, you will find two input files in your folder: `sidle-db-full-sequences.qza`, which is the full length database sequences and `sidle-db-taxonomy.qza`, the taxonomy for the sequences.

```
qiime feature-table tabulate-seqs \
 --i-data sidle-db-full-sequences.qza \
 --o-visualization sidle-db-full-sequences.qzv
```

You can view the artifact using qiime2 view. You should find that there are 5647 squences when you check the data.

2.1 Filtering the Database

Database preparation can optionally begin by filtering the database to remove sequences with too many degenerate nucleotides or with taxonomic assignments that will not be used.

Degenerate filtering limits memory consumption throughout. The authors of SMURF[1] recommend filtering the database to remove sequences with more than 3 degenerate nucleotides. This represents about X% of the greengenes 13_8 database at 99%[2] specificity. a(The the RESCRIPt formatted Silva 138 database is filtered to exclude sequences with more than 5 degenerates[3],[4]). Increasing the number of allowed degenerates (the `--p-max-degen` parameter) will allow more sequences through the filter, and may mean more matches in downstream alignment. However, this comes at a substantial increase in the run time and memory needed, since degenerate sequences have to be expanded, meaning more alignments are required.

For this tutorial, we'll start by filtering to remove anything with more than 3 degenerate nucleotides, since this was the recommended threshold in the original algorithm.

```
qiime sidle filter-degenerate-sequences \
 --i-sequences sidle-db-full-sequences.qza \
 --p-max-degen 3 \
 --o-filtered-sequences sidle-db-full-degen-filtered-sequences.qza
```

Try summarizing your database again. There should be about 5400 sequences remaining. How many do you have?

Some users may also want to filter out sequences which may not be relevant to their analysis, for example, mitochondria or chloroplasts or sequences which are undefined at a high taxonomic level. (Phylum or class, for example.) You can learn more about filtering by taxonomy in the QIIME2 tutorial, but as a brief example, we'll show filtering a greengenes database for features missing a phylum (**p__;**) or kingdom(**k__;**) designation.

```
qiime taxa filter-seqs \
 --i-sequences sidle-db-full-degen-filtered-sequences.qza \
 --i-taxonomy sidle-db-taxonomy.qza \
 --p-exclude "p__;,k__;" \
 --p-mode contains \
 --o-filtered-sequences sidle-db-full-degen-filtered-phylum-def-sequences.qza
```

If you summarize the sequences, you should find that you now have 5419 sequences remaining.

For databases with more complicated strings that include taxonomy, it will be necessary to include the level designation to avoid removing taxa which may be undefined at lower levels.

Note: The taxonomic filtering should be considered carefully and pre-filtering should be very permissive. Many common databases lack clear taxonomic resolution at lower taxonomic levels (family, genus, species) and these sequences still provide meaningful information in reconstruction.

Once you have finished pre-filtering, you are ready to start extracting regions.

[1] Fuks, C; Elgart, M; Amir, A; et al (2018) "Combining 16S rRNA gene variable regions enables high-resolution microbial community profiling." *Microbiome*. **6**:17. doi: 10.1186/s40168-017-0396-x

[2] McDonald, D; Price, NM; Goodrich, J, et al (2012). "An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea." *ISME J*. **6**: 610. doi: 10.1038/ismej.2011.139

[3] Quast, C.; Pruesse, E; Yilmaz, P; et al. (2013) "The SILVA ribosomal RNA gene database project: improved data processing and web-based tools." *Nucleic Acids Research*. **41**:D560. doi: 10.1093/nar/gks1219

[4] Michael S Robeson II, Devon R O'Rourke, Benjamin D Kaehler, et al. "RESCRIPt: Reproducible sequence taxonomy reference database management for the masses."" bioRxiv 2020.10.05.326504; doi: 10.1101/2020.10.05.326504

2.2 Prepare a regional database

The next step is to extract a region of the database. Alignment with the SMURF algorithm relies on extracting the exact kmer to be aligned with your ASVs, so the primer pair and read length must match exactly. Unlike other techniques, there is, unfortunately, no "good enough" approach. To maximize memory efficiency, the database is also prepared by expanding degenerate nucleotides and collapsing duplicated kmers into a single sequence.

First, the region is extracted from the pre-filtered database using the `extract-reads` function from the feature classifier plugin. As an example, we'll look at extracting a region between 316F and 484R using the second primer pair from the SMURF paper (5'-`TCCTACGGGAGGCAGCAG`-3') and (5'-`TATTACCGCGGCTGCTGG`-3').

```
qiime feature-classifier extract-reads \
 --i-sequences sidle-db-full-degen-filtered-phylum-def-sequences.qza \
 --p-f-primer TCCTACGGGAGGCAGCAG \
 --p-r-primer TATTACCGCGGCTGCTGG \
 --o-reads sidle-db-filt-jl.qza
```

For this example, we used the default settings, although these are slightly different from the original SMURF algorithm: In QIIME, the primers are extracted if they have at least an 80% match with the sequence by default; the Matlab implementation of SMURF used a maximum difference of 2 nucleotides[1]. If you wish to use a limit closer to the original algorithm, this can be changed using the `--p-identity` parameter; however, for the sake of this tutorial, we'll use the defaults.

Once the reads have been extracted, they need to be prepared to be used in alignment. This step will expand any degenerate reads that have been extracted, collapse duplicate reads, and trim them to a consistent length. For the full pipeline to work correctly, the primers need to be specified in this step, so once again, you'll need to pass your primers. You'll also need to specify a trim length; let's use 100nt. Finally, we need to specify a regional identifier in the database using the `--region` parameter. This should be the same regional parameter that you use during alignment. We'll call it "WonderWoman" because (a) Diana Prince is amazing and (b) the regional name doesn't matter.

```
qiime sidle prepare-extracted-region \
 --i-sequences sidle-db-filt-jl.qza \
 --p-region "WonderWoman" \
 --p-fwd-primer TCCTACGGGAGGCAGCAG \
 --p-rev-primer TATTACCGCGGCTGCTGG \
 --p-trim-length 100 \
 --o-collapsed-kmers sidle-db-wonder-woman-100nt-kmers.qza \
 --o-kmer-map sidle-db-wonder-woman-100nt-map.qza
```

The command will output the sequences (`--o-collapsed-kmers`) with degenerate sequences expanded and duplicated sequences removed and a mapping between the original sequence name and the kmer name (`--o-kmer-map`). You can use qiime to visualize your kmer map, which gives you the relationship between the original database sequence name (**db-seq**), an expanded name which accounts for degenerates (**seq-name**), the collapsed regional identifier (**kmer**), the primers (**fwd-primer** and **rev-primer**), the region identifier (**region**), and the sequence length (**trim-length**).

```
qiime metadata tabulate \
 --m-input-file sidle-db-wonder-woman-100nt-map.qza \
 --o-visualization sidle-db-wonder-woman-100nt-map.qzv
```

In some cases, the reference region and sequencing length may not be long enough to cover the full amplicon. If that's the case, you can extract the read starting from the reverse primer by setting the trim length to a negative value. You can even reverse complement the resultant amplicons using the `--reverse_complement_result` flag. Let's do an example using the same primers as before, but call the region "Batman".

```
qiime sidle prepare-extracted-region \
 --i-sequences sidle-db-filt-jl.qza \
 --p-region "Batman" \
 --p-fwd-primer TATTACCGCGGCTGCTGG \
 --p-rev-primer TCCTACGGGAGGCAGCAG \
 --p-trim-length -100 \
 --p-reverse-complement-result \
 --o-collapsed-kmers sidle-db-batman-100nt-kmers.qza \
 --o-kmer-map sidle-db-batman-100nt-map.qza
```

As an exercise, try using the 486-650 primers (3-CAGCAGCCGCGGTAATAC-5 forward; 3-CGCATTTCACCGCTACAC-5 reverse) to extract a 100nt region called "GreenLantern". Use the same naming convention as the other two extracted regions (`sidle-db-green-lantern-100nt-kmers.qza`).

Now, you have a database that's ready to use for alignment and reconstruction.

2.3 TL;DR: Database Preparation

2.3.1 Database Filtering

- Filtering only needs to be performed once per dataset
- Degenerate filtering speeds up preparation and alignment
- You can exclude sequences during database generation that you don't want included in the final table

Degenerate Filtering

Syntax

```
qiime sidle filter-degenerate-sequences \
 --i-sequences [unfiltered sequences].qza \
 --p-max-degen [degenerate threshold] \
 --o-filtered-sequences [filtered sequences].qza
```

Example

```
qiime sidle filter-degenerate-sequences \
 --i-sequences sidle-db-full-sequences.qza \
 --p-max-degen 3 \
 --o-filtered-sequences sidle-db-full-degen-filtered-sequences.qza
```

Taxonomic Filtering

Please see the qiime filtering tutorial for more information.

Syntax

```
qiime taxa filter-seqs \
 --i-sequences [unfiltered sequences].qza \
 --i-taxonomy [taxonomic descriptions].qza \
 --p-exclude [criteria to exclude] \
 --p-mode contains \
 --o-filtered-sequences [filtered sequences].qza
```

Example

```
qiime taxa filter-seqs \
 --i-sequences 85_otus-filtered.qza \
 --i-taxonomy ref-taxonomy.qza \
 --p-exclude "p__;,k__;" \
 --p-mode contains \
 --o-filtered-sequences 85-otus-filtered-defined-phylum.qza
```

2.3.2 Database Region Preparation

- The primers used to extract regions must be the same as the primers used to amplify your sequences in that region
- The extraction command must be re-run for each primer pair and database
- Read preparation needs to be re-run for each primer pair, read length, and database
- A negative trim length to `qiime sidle prepare-extracted-region` will trim from the reverse primer (right)

Read Extraction

Please see the qiime feature classifier documentation for more information.

Syntax

```
qiime feature-classifier extract-reads \
 --i-sequences [full length sequences] \
 --p-f-primer [forward primer] \
 --p-r-primer [reverse primer] \
 --o-reads [extracted region]
```

Example

```
qiime feature-classifier extract-reads \
 --i-sequences 85-otus-filtered-defined-phylum.qza \
 --p-f-primer TGGCGGACGGGTGAGTAA \
 --p-r-primer CTGCTGCCTCCCGTAGGA \
 --o-reads 85-outs-filtered-defined-phylum-extract-jl.qza
```

Regional Database Preparation

Syntax

```
qiime sidle prepare-extracted-region \
 --i-sequences [extracted sequences].qza \
 --p-region [region label] \
 --p-fwd-primer [forward primer for region] \
 --p-rev-primer [reverse primer for region] \
 --p-trim-length [kmer length] \
 --o-collapsed-kmers [kmer sequences].qza \
 --o-kmer-map [kmer to database map].qza
```

Example

For forward reads (trim from the left)

```
qiime sidle prepare-extracted-region \
 --i-sequences sidle-db-filt-jl.qza \
 --p-region "WonderWoman" \
 --p-fwd-primer TCCTACGGGAGGCAGCAG \
 --p-rev-primer TATTACCGCGGCTGCTGG \
 --p-trim-length 100 \
 --o-collapsed-kmers sidle-db-wonder-woman-100nt-kmers.qza \
 --o-kmer-map sidle-db-wonder-woman-100nt-map.qza
```

For reverse reads (trim from the right and in this case, reverse complement). The primers should be flipped (we'll trim from the forward primer)

```
qiime sidle prepare-extracted-region \
 --i-sequences sidle-db-filt-jl.qza \
 --p-region "Batman" \
 --p-fwd-primer TATTACCGCGGCTGCTGG \
 --p-rev-primer TCCTACGGGAGGCAGCAG \
 --p-trim-length -100 \
 --p-reverse-complement-result \
 --o-collapsed-kmers sidle-db-batmap-100nt-kmers.qzv \
 --o-kmer-map sidle-db-batman-100nt-map.qzv
```

Database References

CHAPTER 3

Read Preparation

Although the original SMURF paper relied on a quality filtering protocol, we have elected to recommend the use of existing denoising algorithms. In Sidle, we recommend using existing tools to perform pre-processing. Here, we provide example code for processing samples, however, this can be accomplished any number of ways.

Note: The following steps represent a **suggested** pipeline for preparing reads to be used with sidle. Alternative pipelines are possible and may be more useful for your specific circumstances.

As an initial example of read preparation, your pipeline may include the following steps. To make navigation easy, what stage are you at?

- **I don't know. Someone handed me files and told me to analyze them!**
  - Contact your sequencing provider or the person the files came from if they were sequenced locally
  - Check the repository where they came from. EBI and SRA typically provide files *demultiplexed by sample*; Qiita provides *deblured tables*
- **I have multiplexed sequences (multiple samples in the same fastq file)**
  - You need to *demultiplex by sample and region*
  - Then, you want to *denoise the sequences*
- **I have two fastq files per sample**
  - Your sequences are already demultiplexed by samples; you need to *demultiplex by region*
  - Then, you want to *denoise the sequences*
- **I have two fastq files per sample and per region**
  - Your sequences are already demultiplexed. Make you check you've *demultiplexed correctly*
  - Then, you need to *denoise the sequences*
- **I have an ASV table for each region and a corresponding representative sequence file**
  - It was *denoised with dada2*

- It was *denoised with deblur*

3.1 Demultiplex your reads

3.1.1 Fully Multiplexed sequences

The first step of sample processing is demultiplexing your sequences into sample x region pairs. You may have fully multiplexed sequences, in which case, you will need to demultiplex into both samples and regions. If this is the case, you will likely have two or three fastq files, which likely represent forward, reverse, and index reads. Please refer to the QIIME 2 documentation for demultiplexing EMP sequences and demultiplexing with cutadapt.

Now, you're ready to *denoise the regions*.

3.1.2 Sequences barcoded by sample (mixed regions per sample)

If your samples are demultiplexed to include a single set of files (probably forward and reverse), you will need to demultiplex the files into regions. You can use the cutadapt[1] plugin in QIIME 2 to demultiplex into regions and remove the primers in the same step. If you have paired-end reads, you will want to use the `trim-paired` command; if you have Ion-Torrent or are using a subset of reads, you might choose to use `trim-single`. (See the q2 cutadapt and cutadapt docs for more details.) Using the `--p-discard-untrimmed` flag will remove any sequence which does not have the primer region, allowing you to find our sequences of interest.

This will give you a table per region. Next, continue on to *denoise the regions*.

3.1.3 Sequences barcoded by sample and region

If you are lucky, or planned ahead, you may have added a unique barcode to each region of a sample. So, if, for example, you sequenced 5 samples with 6 regions, after demultiplexing, you should have 30 demultiplexed files and a map which describes the relationship between the name and region.

You'll need that later. We suggest a file with at least three columns: `sample-id`, `original-sample-id`, and `regional-id`. (Note that `sample-id` is a QIIME-required header column.) You could potentially combine this with a manifest or barcode file that you used to import and/or demultiplex your data. (If you don't if your data is demultiplexed already, check in with your sequencing center! Make friends with your sequencing center. Invite them to drink a hot beverage with you and talk cool research.)

Depending on your region lengths and sequencing length, you may need to map the forward and reverse reads separately. So, you may need final sets of demultiplexed reads which represent the

- paired-end reads for regions which can be merged
- forward reads only for regions which cannot be merged
- reverse reads only for regions which cannot be merged

Note: If you have a region which does not overlap, the forward and reverse reads should be processed as separate kmers and then reconstructed, rather than concatenated during denoising.

Next, make sure that you trim your primers. If you plan to denoise all the data together, this should be done sequentially for each primer pair that you plan to use for each block of sequences. (i.e. you should trim the paired reads, forward reads, and reverse reads separately.)

[1] Martin, M. (2011). "Cutadapt removes adapter sequences from high-throughput sequencing reads". *EMBnet.journal* **17**:10. doi: https://doi.org/10.14806/ej.17.1.200

This will give you a table where each sample is named whatever you've linked to your barcode. From here, you can either filter your sequences before denoising, or proceed combining all regions, and then filter the table later. If you denoise with Dada2, you may find better performance if you leave the sequences together; this will not affect denoising with deblur.

3.2 Denoise reads with your favorite algorithm

Reads should be denoised, possibly merged, and *trimmed* to a standard length for kmer-based alignment. Depending on the algorithmic approach, Dada2[2] and Deblur[3] might both be good approaches. (If you're interested in an independent comparison of the two methods outside reconstruction, Nearing et al provides an independent benchmark[4]).

However, there are some limitations. Ion Torrent and 454 pyrosequencing results should be *denoised with dada2*. Dada2 tends to retain more high quality sequences than deblur, but may inflate the number of features. Because of the algorithm, it also has a longer run time.

Illumina data which has already been joined or quality filtered should be *denoised with deblur*. It's a faster algorithm and highly parallelizable but it's also more conservative.

3.2.1 DADA2

There are several helpful tutorials on the QIIME 2 website that describe running dada2 on forward reads and dada2 on paired reads. Minimal pre-processing should be applied before DADA2: simply demultiplex your data and pass it into the command.

Once DADA2 has been run, you will need to trim the reads to a consistent length. This can be done using the qiime dada2 parameters during denoising, or with the `trim-dada2-posthoc` method in q2-sidle.

As an example of the command, we can download the feature table and representative sequences from the qiime2 Moving Pictures Tutorial and then practice.

```
wget https://docs.qiime2.org/2020.6/data/tutorials/moving-pictures/table-dada2.qza .
wget https://docs.qiime2.org/2020.6/data/tutorials/moving-pictures/rep-seqs-dada2.qza␣
↪.
```

If you look at the sequence summary (viewable here), you'll find the sequences have already been trimmed to 120nt. However, for the alignment we plan to do, it may be useful to trim them to 100nt.

```
qiime sidle trim-dada2-posthoc \
 --i-table table-dada2.qza \
 --i-representative-sequences rep-seqs-dada2.qza \
 --p-trim-length 100 \
 --o-trimmed-table table-dada2-100nt.qza \
 --o-trimmed-representative-sequences rep-seq-dada2-100nt.qza
```

You can check the length by tabulating the sequences.

```
qiime feature-table tabulate-seqs \
 --i-data rep-seq-dada2-100nt.qza \
 --o-visualization rep-seq-dada2-100nt.qzv
```

[2] Callahan, B; McMurdie, P; Rosen, M; et al (2016) "Dada2: High resolution sample inference from Illumina amplicon dada." *Nature Methods*. **13**: 581. doi: https://doi.org/10.1038/nmeth.3869

[3] Amir, A; McDonald, D; Navas-Molina, JA et al. (2017) "Deblur Rapidly Resolves Single-Nucleotide Community Sequence Patterns". *mSystems*. **2**:e00191 doi: 10.1128/mSystems.00191-16

[4] Nearing, J.T.; Douglas, G.M.; Comeau, A.M.; Langille, M.G.I. (2018) "Denoising the Denoisers: an independent evaluation of microbiome sequencing error-correction approaches." *Peer J*. **6**: e5364 doi: 10.7717/peerj.5364

You should find the sequences all trimmed to 100nt, and ready for alignment.

3.2.2 Deblur

If you have sequenced using Illumina, Deblur may be easier to use and is recommended by the authors/original developers of SMURF. You can find a tutorial for deblurring single end reads or paired end reads on the QIIME webpage. Simply set your Deblur trim length to the final kmer length you'll use and proceed.

3.3 Check your tables

Before you proceed, make sure that you have what you need. For alignment and reconstruction to work correctly, you will need one feature table and one representative sequence set for each region you plan to align. The ASVs in a feature table should have a consistent length. All the samples in the table should have the same names.

If you need to, *trim all the ASVs in your table to a consistent length* or rename your samples.

3.4 TL;DR: Read Preparation

A quick flowchart for figuring out how to demultiplex and pre-process your reads.

3.4.1 Demultiplexing

- **You need to determine if your reads have already been multiplexed and how, and import/demultiplex accordingly**
  - EMP Demultiplexing
  - Cutadapt Demultiplexing
  - Import already demultiplexed reads into QIIME 2
- Samples with mixed regions can be extracted using cutadapt to trim primers and discard untrimmed reads.

Paired End Command

Single End Command

```
qiime cutadapt trim-single \
 --i-demultiplexed-sequences all_regions_fwd.qza \
 --p-front TCCTACGGGAGGCAGCAG \
 --p-discard-untrimmed \
 --p-error-rate 0.15 \
 --o-trimmed-sequences trimmed-regions/316_484_fwd_demux.qza
```

3.4.2 Denoising and Table Preparation

- **See the relevant QIIME 2 tutorials:**
  - **Dada2**
    - Single end reads: Moving Pictures Option 1
    - Paired end reads: Atacama Soils Tutorial

  - **Deblur**
    - Single end reads: Moving Pictures Option 2
    - Paired end reads: Alternative Methods of Read Joining Tutorial
- Make sure to trim your sequences to the same length that was used for your database. You can do this with the `trim-dada2-posthoc` command.

Syntax

```
qiime sidle trim-dada2-posthoc \
 --i-table [table filepath] \
 --i-representative-sequences [sequence filepath] \
 --p-trim-length [trim length] \
 --o-trimmed-table [trimmed table] \
 --o-trimmed-representative-sequences [trimmed sequences]
```

Example

```
qiime sidle trim-dada2-posthoc \
 --i-table table-dada2.qza \
 --i-representative-sequences rep-seqs-dada2.qza \
 --p-trim-length 100 \
 --o-trimmed-table table-dada2-100nt.qza \
 --o-trimmed-representative-sequences rep-seq-dada2-100nt.qza
```

Read Preparation References

CHAPTER 4

Sequence Reconstruction

The core of the SMURF algorithm is based on the kmer-based reconstruction of short regions into a full-length framework. Within Sidle, there are two steps in database reconstruction. First, ASVs are aligned on a regional basis to generate the local kmer-based alignment. Then, the full collection of sequences is assembled into a reconstructed table of counts. For this example, we'll work with a small, entirely artifical subset of samples that are designed to run quickly.

If you've already done the database tutorial, make sure that you're in the `sidle_tutorial` directory.

```
pwd
```

If you're new to the tutorial, you can make a new tutorila directory by running

```
mkdir -p sidle_tutorial
cd sidle-tutorial
```

Next, you will need to get the tutorial data. You will need to download three sets of files: the sequencing data, the alignments, and the reconstruction files.

If you have not run the database tutorial, you will also want to get the database data.

```
wget https://github.com/jwdebelius/q2-sidle/blob/main/docs/tutorial_data/database.tgz
tar -xzf database.tgz
```

4.1 Regional alignment

The first step in reconstruction is to perform per-region alignment between the sequences and the database. We'll do this with the `align-regional-kmers` command. We set the reference database that we extracted previously as the `--kmer-db-fp`. The ASV represent sequences and are passed as the `--rep-seq-fp`. Finally, we supply a regional definition. This should be the same as the region name that you gave when you extracted the kmers. In this case, the region name was "WonderWoman".

Alignment is a pleasantly parallelizable problem, meaning that we can get improved performance by distributing the jobs. You can set this in almost any `sidle` command using the `--p-n-workers` parameter. (And you can learn

more about parallel processing doc:*here <parallel_processing>*). For this example, we'll use 2 cores; if you have more avaliable you can or should use them.

```
   qiime sidle align-regional-kmers \
--i-kmers database/sidle-db-wonder-woman-100nt-kmers.qza \
--i-rep-seq data/wonder-woman-100nt-rep-set.qza \
--p-region WonderWoman \
--p-n-workers 2 \
--o-regional-alignment alignment/wonder-woman-align-map.qza
```

This will output an alignment file and any ASV sequences which wouldn't be aligned to the database, for your own record keeping.

Note: If you are unsure of the region name or read length for your database kmers, you can always check the provenance by visiting view.qiime2.org

Optionally, you can also modify parameters for the number of basepairs that differ between the reference and representative sequences (`--p-max-mismatch`); the original paper uses a mismatch of 2 with 130nt sequences.

You may find that if you have longer kmers, you might want to increase this parameter accordingly. A lower (more stringent) value will increase the number of discarded sequences, while a higher number may mean your matches are lower quality. You may find that if you have longer kmers, you may want to increase this parameter accordingly. A lower (more stringent) value will increase the number of discarded sequences, a higher number may mean your matches are lower quality.

Using the same parameters, you will need to align the other two regions.

```
qiime sidle align-regional-kmers \
 --i-kmers database/sidle-db-batman-100nt-kmers.qza \
 --i-rep-seq data/batman-100nt-rep-set.qza \
 --p-region Batman \
 --p-n-workers 2 \
 --o-regional-alignment alignment/batman-align-map.qza

qiime sidle align-regional-kmers \
 --i-kmers alignment/green-lantern-kmer-db.qza \
 --i-rep-seq table/green-lantern-rep-seq.qza \
 --p-region GreenLantern \
 --p-n-workers 2 \
 --o-regional-alignment alignment/green-lantern-align-map.qza
```

Now, you have all three local alignments prepared, you're ready to reconstruct your table.

4.2 Table Reconstruction

The table is reconstucted in three steps. First, the regional fragments get re-assembled into complete database sequences. Then, the relative abundance of the pooled counts gets computed through an optimization process. Finally, the relative abundance is used to reconstruct a table of counts.

The `per-nucleotide-error` is combined with the `maximum-mismatch` parameter from alignment to the probability that a sequence that differs from the reference. So, for instance, this algorithm allows a single ASV to be mapped to multiple sequences in the reference database. During reconstruction, the alignment mismatch, sequencing error, and relative abundance are combined to calculate the mapped abundance.

The `min-abundance` determines the relative abundance of a database sequence to be excluded during optimization. This is, to some degree, a function of the avaliable sequencing depth and the desired specificity of the fit.

Finally, let's plan on running the command in parallel, using the `--p-n-workers` flag; this is particularly useful in the per-sample reconstruction step. We'll use 2 workers in this tutorial, if you have more avaliable you may prefer that.

Now, let's reconstruct the table, using the default settings.

```
qiime sidle reconstruct-counts \
 --p-region WonderWoman \
  --i-kmer-map database/sidle-db-wonder-woman-100nt-map.qza \
  --i-regional-alignment alignment/wonder-woman-align-map.qza \
  --i-regional-table data/wonder-woman-100nt-table.qza \
 --p-region Batman \
  --i-kmer-map database/sidle-db-batman-100nt-map.qza \
  --i-regional-alignment alignment/batman-align-map.qza \
  --i-regional-table data/batman-100nt-table.qza \
 --p-region GreenLantern \
  --i-kmer-map database/sidle-db-batman-100nt-map.qza \
  --i-regional-alignment alignment/green-lantern-align-map.qza \
      --i-regional-table data/green-lantern-100nt-table.qza \
     --p-n-workers 2 \
 --o-reconstructed-table reconstruction/league_table.qza \
 --o-reconstruction-summary reconstruction/league_summary.qza \
 --o-reconstruction-map reconstruction/league_map.qza
```

The command will produce a count table, a file containing details about the number of database kmers mapped to a region along with the ASV IDs, and a mapping that's needed if you want to do taxonomic reconstruction.

Let's take a look at the count table.

```
qiime feature-table summarize \
 --i-table reconstruction/league_table.qza \
 --o-visualization reconstruction/league_table.qzv
```

You'll notice that some of the feature IDs contain a `|` character, for example, `1764594|195532|4471854`. This means the two databases sequences could not be resolved during the reconstruction, and so we assign the sequence to both regions. The more regions that are used in the reconstruction, the more likely you are to be able to accurately reconstuct the database sequences.

The second output is a summary. The summary can be used to evaluate the quality of the reconstruction; see the original manuscript[1] for more details. By default, the summary will consider degenerate kmers as unique sequences; you can change the behavior using the `count-degenerates` parameter; when False, kmers will only be counted if they belong to unique reference sequences. You can view the summary by tabulating the metadata.

```
qiime metadata tabulate \
 --m-input-file reconstruction/league_summary.qza \
 --o-visualization reconstruction/league_summary.qzv
```

Let's look at the information for the unresolved feature, `1764594|195532|4471854`. How many regions has it found?

[1] Fuks, C; Elgart, M; Amir, A; et al (2018) "Combining 16S rRNA gene variable regions enables high-resolution microbial community profiling." *Microbiome*. **6**:17. doi: 10.1186/s40168-017-0396-x

4.3 Taxonomic Reconstruction

Now you have the table reconstructed, you're ready to reconstruct the taxonomy to match. Specifcially, this process addresses cases where multiple database sequences cannot be untangled. The function takes the database map generated during reconstruction and the taxonomy associated with the database, and returns the reconstructed taxonomy.

There are three possible general cases for a set of shared sequences. First, they can share the full taxonomic string; second, they may differ at some point: or third, they may be same until one is missing an assignment. Let's start with the simplest case. If we have two database sequences:

```
1234    k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Blautia; s__obeum
1235    k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Blautia; s__obeum
```

Then, when we reconstruct taxonomy, everything is the same and the final taxonomic label should be:

```
1234 | 1235 k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Blautia; s__obeum
```

There's also thee possibility that sequences differ at some higher level, for example:

```
1236    k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Blautia; s__obeum
1237    k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Roseburia; s__
```

In that case, the algorithm would keep the taxonomic assignment associated with the most recent common ancestor:

```
1236 | 1237 k__Bacteria; p__Firmictues; c__Clostridia; o__Clostridiales; f__
↪Lachnospiraceae; g__Blautia | g__Roseburia; g__Blautia | g__Rosburia
```

The `--database` parameter allows the user to select the type of database being used (`greengenes`, `silva` or `none`). If the database is a defined database(`greengenes` or `silva`), some ad-hoc database cleaning will be performed automatically , specifically with regard to the `define-missing` and `ambiguity-handling` parameters. For example, if a defined string is:

```
k__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Entrobacteriales; f__
↪Enterobacteriaceae; g__; s__
```

Then, the new, cleaned string will be:

```
k__Bacteria; p__Proteobacteria; c__Gammaproteobacteria; o__Entrobacteriales; f__
↪Enterobacteriaceae; g__unsp. f. Enterobacteriaceae; s__unsp. f. Enterobacteriaceae
```

Our database is a subset of the greengenes database, so let's specify that we used the greengenes database and inherit the missing strings.

```
qiime sidle reconstruct-taxonomy \
 --i-reconstruction-map reconstruction/league_map.qza \
 --i-taxonomy database/sidle-db-taxonomy.qza \
 --p-database 'greengenes' \
 --p-define-missing 'inherit' \
 --o-reconstructed-taxonomy reconstruction/league_taxonomy.qza
```

You can check the taxonomic reconstruction by tabulating the taxonomy.

```
qiime metadata tabulate \
 --m-input-file reconstruction/league_taxonomy.qza \
 --o-visualization reconstruction/league_taxonomy.qzv
```

What's the taxonomy assignment for `1764594|195532|4471854`?

4.4 Reconstructing the Phylogenetic Tree

The last step in reconstruction is to reconstruct fragments for the phylogenetic tree. Unfortunately, if the reference sequences cannot be resolved, the phylogenetic tree cannot simply be inherited from the database. So, we need to reconstruct a new phylognetic tree. We handle sequences in two ways.

1. Any database sequence which could full resolved can keep it's position in the reference tree
2. Sequences which can't be resolved need to handled somehow.

We could randomly select a sequence to map the reconstructed region to. However, that might not work when there are several sequences that got combined. So, instead, if we can't resolve the database sequence, we calculate a concensus sequence from the combined data, extract them over the regions we were able to map, and then those consensus sequences can be inserted into a phylogenetic reference backbone using SEPP or something similar.

Note: Sucessful reconstruction requires that the ids in the database you used as your reference for reconstruction and the database you're using for alignment are the same. Make sure that you are using the same database release version and the same level of sequence identity.

So, our first step is to reconstruct the consensus fragments from sequences that could not be resolved.

```
qiime sidle reconstruct-fragment-rep-seqs \
--p-region WonderWoman \
  --i-regional-alignment alignment/wonder-woman-align-map.qza \
 --p-region Batman \
  --i-regional-alignment alignment/batman-align-map.qza \
 --p-region GreenLantern \
  --i-regional-alignment alignment/green-lantern-align-map.qza \
 --i-reconstruction-map reconstruction/league_map.qza \
 --i-reconstruction-summary reconstruction/league_summary.qza \
 --i-aligned-sequences database/sidle-db-aligned-sequences.qza \
 --o-representative-fragments reconstruction/league-rep-seq-fragments.qza
```

We can then insert the sequences into the reference tree. Let's first get the reference tree.

```
wget \
 -O "sepp-refs-gg-13-8.qza" \
 "https://data.qiime2.org/2020.11/common/sepp-refs-gg-13-8.qza"
```

Then, we'll do the fragment insertion.

```
qiime fragment-insertion sepp \
 --i-representative-sequences reconstruction/league-rep-seq-fragments.qza \
 --i-reference-database sepp-refs-gg-13-8.qza \
 --o-tree reconstruction/league-tree.qza \
 --o-placements reconstruction/league-placements.qza
```

Now, you're ready to analyze your data.

4.5 Next Steps: Analysis!

You now have a reconstructed table, and associated taxonomy. Go forth and enjoy your analysis. The QIIME 2 tutorials offer some good options of downstream diversity and statistical analyses that can be done with this data.

4.6 TL;DR Reconstruction

4.6.1 Regional Alignment Commands

- The region name for the alignment **must match** the region name used for building the kmer map
- Kmers and representative sequences must be the same length
- This step is performed on a per-region basis

Syntax

```
qiime sidle align-regional-kmers \
 --i-kmers [kmer sequences from extracted database] \
 --i-rep-seq [ASV representative sequnces] \
 --p-region [Region name] \
 --o-regional-alignment [regional alignment]
```

Example

```
qiime sidle align-regional-kmers \
 --i-kmers wonderwoman-kmer-db.qza \
 --i-rep-seq wonderwoman-rep-seq.qza \
 --p-region WonderWoman \
 --o-regional-alignment wonderwoman-align-map.qza
```

4.6.2 Reconstructing the Table

- Make sure your region names match between the alignment artifact, the database kmer map, and the `region` parameter.
- `count-degenerates` will control how the summary describes differences in the sequences
- `region-normalize` will affect how many counts are assigned in the final table

Syntax

For *n* regions

```
qiime sidle reconstruct-counts \
 --p-region [region 1 name] \
  --i-kmer-map [region 1 kmer map] \
  --i-regional-alignment [region 1 alignment] \
  --i-regional-table [region 1 counts table] \
  ... \
  --p-region [region n name] \
  --i-kmer-map [region n kmer map] \
  --i-regional-alignment [region n alignment] \
  --i-regional-table [region n counts table] \
 --o-reconstructed-table [reconstructed table] \
```

(continues on next page)

(continued from previous page)

```
 --o-reconstruction-summary [reconstruction summary] \
 --o-reconstruction-map [reconstructed database map]
```

Example

```
qiime sidle reconstruct-counts \
 --p-region WonderWoman \
  --i-kmer-map database/sidle-db-wonder-woman-100nt-map.qza \
  --i-regional-alignment alignment/wonder-woman-align-map.qza \
  --i-regional-table data/data/wonder-woman-100nt-table.qza \
 --p-region Batman \
  --i-kmer-map database/sidle-db-batman-100nt-map.qza \
  --i-regional-alignment alignment/batman-align-map.qza \
  --i-regional-table data/batman-100nt-table.qza \
 --p-region GreenLantern \
  --i-kmer-map database/sidle-db-green-lantern-100nt-map.qza \
  --i-regional-alignment alignment/green-lantern-align-map.qza \
  --i-regional-table data/green-lantern-100nt-table.qza \
 --o-reconstructed-table reconstruction/league_table.qza \
 --o-reconstruction-summary reconstruction/league_summary.qza \
 --o-reconstruction-map reconstruction/league_map.qza
```

4.6.3 Reconstructing taxonomy

- A database specification is required

Syntax

```
qiime sidle reconstruct-taxonomy \
 --i-reconstruction-map [reconstruction map] \
 --i-taxonomy [taxonomy path] \
 --p-database [database name] \
 --o-reconstructed-taxonomy [reconstructed taxonomy]
```

Example

```
qiime sidle reconstruct-taxonomy \
 --i-reconstruction-map reconstruction/league_map.qza \
 --i-taxonomy database/sidle-db-taxonomy.qza \
 --p-database 'greengenes' \
 --p-define-missing 'inherit' \
 --o-reconstructed-taxonomy reconstruction/league_taxonomy.qza
```

4.6.4 Reconstructing the Tree

- A phylogenetic tree can be reconstructed by first estimating the consensus fragments for the original sequences and then inserting them into a tree.
- See the q2-fragment-insertion documentation for more information

Fragment reconstruction syntax

```
qiime sidle reconstruct-fragment-rep-seqs \
 --i-reconstruction-map [reconstruction map] \
```

(continues on next page)

(continued from previous page)

```
 --i-reconstruction-summary [reconstruction summary] \
 --i-aligned-sequences [aligned sequences] \
 --m-manifest-file [manifest] \
 --o-representative-fragments [concensus fragments]
```

Example reconstruction syntax

```
qiime sidle reconstruct-fragment-rep-seqs \
 --i-reconstruction-map reconstruction/league_map.qza \
 --i-reconstruction-summary reconstruction/league_summary.qza \
 --i-aligned-sequences database/sidle-db-aligned-sequences.qza \
 --m-manifest-file manifest.txt \
 --o-representative-fragments reconstruction/league-rep-seq-fragments.qza
```

4.6.5 References

CHAPTER 5

Parallel Processing in Sidle

Sidle is built to be run in parallel using dask. This system allows for flexible implementations. All sidle commands except `reconstruct taxonomy` allow parallel processing. There are three ways to manage parallel processing:

- `--debug` which turns off all parallel processing. This is primarily implemented for testing
- `--p-n-workers` will create a new dask cluster with the specified number of workers and will use the avalaible resources
- `--p-client-address` allows you to pass in a pre-configured cluster for processing. You can learn more about setting up dask clients in their documentation.

This hopefully provides a flexible interface for parallel procesisng of your samples

CHAPTER 6

Indices and tables

- genindex
- modindex
- search