

Claident を用いた定量メタバーコーディング解析

田辺晶史 (東北大学大学院生命科学研究科)

2024-01-22

1 Claident を用いた定量メタバーコーディング解析

Claident は、筆者が開発・メンテナンスしている、メタバーコーディングや DNA バーコーディングのための塩基配列データ解析プログラム集です。MiFish pipeline (Sato et al. 2018, Zhu et al. 2023) との違いは、大まかには以下の通りです。

- MiFish プライマー (Miya et al. 2015, 2020) を用いた魚類メタバーコードデータだけでなく、全生物・ウィルスのあらゆる遺伝子座のデータに対応
- 定量・非定量メタバーコーディング (Ushio et al. 2018a) をサポート
- より柔軟で詳細な解析に対応
- Web サービスではなく、自前のコンピュータで解析を行う
- 使用のための前提知識・必要な物品は多い

ここでは、Claident のインストールから内部標準 DNA を利用した定量メタバーコーディングの方法を解説します。Claident の詳細については下記 URL をご参照下さい。

<https://www.claident.org/>

以下では、Linux・macOS のターミナル環境での作業に習熟している方向けに解説を行っていきます。ターミナル環境での作業に不慣れな方は、予め習得しておく必要があります。

1.1 Claident の動作環境およびインストール方法

Claident は、以下の環境で動作するように作成されています。

- Debian 11 以降
- Ubuntu 20.04 以降 (Windows 上の WSL2 環境を含む)
- Linux Mint 20 以降
- RedHat Enterprise Linux 8 以降
- AlmaLinux 8 以降 (Windows 上の WSL2 環境を含む)
- Rocky Linux 8 以降

- Homebrew をインストールした macOS
- MacPorts をインストールした macOS

Windows をご使用の方は、Microsoft Store から「Windows Subsystem for Linux」、「Ubuntu」および「Windows Terminal」をインストールすれば、Ubuntu 環境内に Claident をインストールすることができます。ただし、Windows 上にインストールした Ubuntu は、標準では最大 250GB 程度しかディスク容量を使用できません (執筆時点)。大きなデータ解析にはディスク容量が不足する可能性が高いので、専用の解析マシンを用意することをお勧めします。分子同定の際に大きな参照配列データベースを使用すると膨大なメモリを必要とするため、できるだけメモリを多く搭載したマシンが望ましいでしょう。

Debian・Ubuntu・Linux Mint・Windows 上にインストールした Ubuntu の場合、ターミナル上で以下のコマンドを実行することで Claident をインストールすることができます。

```
sudo apt install wget
mkdir temporary
cd temporary
wget https://www.claident.org/installClaident_Debian.sh
wget https://www.claident.org/installOptions_Debian.sh
wget https://www.claident.org/installUCHIMEDB_Debian.sh
wget https://www.claident.org/installlDB_Debian.sh
sh installClaident_Debian.sh
sh installOptions_Debian.sh
sh installUCHIMEDB_Debian.sh
sh installlDB_Debian.sh
cd ..
rm -rf temporary
```

Homebrew をインストールした macOS で Claident をインストールするには、ターミナル上で以下のコマンドを実行します。

```
brew install wget
mkdir temporary
cd temporary
wget https://www.claident.org/installClaident_macOSHomebrew.sh
wget https://www.claident.org/installOptions_macOSHomebrew.sh
wget https://www.claident.org/installUCHIMEDB_macOSHomebrew.sh
wget https://www.claident.org/installlDB_macOSHomebrew.sh
sh installClaident_macOSHomebrew.sh
sh installOptions_macOSHomebrew.sh
sh installUCHIMEDB_macOSHomebrew.sh
sh installlDB_macOSHomebrew.sh
cd ..
rm -rf temporary
```

なお、ファイアーウォールの内側など、プロキシサーバを通してしか外部ネットワークにアクセスできない環境では、以下のコマンドをターミナル上で実行してから前述のインストールコマンドを実行する必要があります。

```
export http_proxy=http://proxyaddress:portnumber
export https_proxy=http://proxyaddress:portnumber
export ftp_proxy=http://proxyaddress:portnumber
```

プロキシサーバがユーザー名とパスワードでの認証を要する場合、上記コマンドの代わりに以下のコマンドを実行します。

```
export http_proxy=http://username:password@proxyaddress:portnumber
export https_proxy=http://username:password@proxyaddress:portnumber
export ftp_proxy=http://username:password@proxyaddress:portnumber
```

前述のインストールコマンドでは、いずれの環境でも `/usr/local` 以下にインストールされますが、インストール先を変更したい場合、インストールコマンド実行前に以下のコマンドを実行します。

```
export PREFIX=/home/tanabe/claident20240101
```

上記の例では、`/home/tanabe/claident20240101` 以下に Claident はインストールされます。インストール先を変更した場合、環境変数 `PATH` に実行コマンドがある **インストール先/bin** が登録されていないため、Claident の解析コマンドが実行できません。そこで、Claident で解析を行う前に以下のコマンドを実行して環境変数 `PATH` に **インストール先/bin** を加えます。

```
export PATH=/home/tanabe/claident20240101/bin:$PATH
```

Claident で解析前に上記コマンドを毎回実行するのが面倒な場合、`~/.bashrc` の末尾などに上記コマンドを記述すると、ターミナル起動時に毎回自動的に実行されるようになります。

このように、インストール先を変更することで、複数のバージョンの Claident を共存させることができます。ただし、Claident の各コマンドは設定ファイル `~/.claident` を参照していますので、使用する Claident を切り替えるには `~/.claident` も変更する必要があります。`.claident` のテンプレートは、**インストール先/share/claident/.claident** に存在していますので、このファイルを `~/.claident` に上書きコピーすれば Claident が完全に切り替わります。実際に複数のバージョンを 1 台のマシンにインストールして共存させる場合、異なるユーザーを作成してそれぞれで Claident をユーザーの所有ディレクトリ内にインストールし、ユーザーを切り替えることで使用する Claident のバージョンを切り替えるようにするのが良いでしょう。

1.2 データ解析全体の流れと前提条件

Claident によるデータ解析は、以下の流れで行います。

1. デマルチプレクシング
2. ペアエンド配列の連結
3. 低品質配列の除去 (Edgar and Flyvbjerg 2015)
4. デノイジング (Callahan et al. 2016)
5. 参照配列データベースを用いないキメラ除去 (Edgar 2016, Rognes et al. 2016)
6. 内部標準配列クラスタリング (Edgar 2010, Rognes et al. 2016)
7. 参照配列データベースを用いたキメラ除去 (Edgar et al. 2011, Rognes et al. 2016)
8. インデックスホッピング除去 (Esling et al. 2015)
9. ネガティブコントロールを利用したデコンタミネーション
10. 分子同定 (Tanabe and Toju 2013)

11. サンプル × OTU 表の作成・加工
12. カバレッジベースレアファクション (Chao and Jost 2012)
13. 内部標準 DNA リード数を利用した DNA 濃度の推定 (Ushio et al. 2018a)

最終的に得られたサンプル × OTU 表を R やその他の統計解析環境で処理することで、作図や要約、仮説検証を行います。Claident 自体には統計解析機能はありません。

Claident は大抵のメタバーコードデータの解析に使用可能ですが、ここでは以下のようなデータを仮定して解説を進めます (下記を満たしていないデータを全く解析できないわけではありません)。

- 環境水を濾過して濾過フィルターから抽出した環境 DNA サンプルとネガティブコントロールとしてのフィールドブランクが含まれる
- 以下の方法でライブラリ調製
 - 濃度のわかっている複数の内部標準 DNA を添加して MiFish プライマーを使用して tailed PCR (1st PCR)
 - 1st PCR 産物を鋳型にしてインデックスプライマーを使用して tailed PCR (2nd PCR)
- 各サンプルの 2nd PCR 産物を混合して Illumina 社製シーケンサで **1 ランまたは 1 レーン専有**で解読

したがって、サンプル・ブランクごとに以下の情報がわかっている必要があります。

- サンプル・ブランクのいずれなのか
- 濾過水量
- 抽出 DNA 溶液量 (回収液量ではなく、最後の溶出時に添加した液量)
- 内部標準 DNA 塩基配列
- 内部標準 DNA 濃度
- 1st PCR 時のプライマー配列のうち、シーケンサの読み始めになる部分配列
- 2nd PCR 時のプライマー配列のうち、インデックスとして読まれる部分配列

フィールドブランクがない、または十分な数がない場合、抽出ブランクや 1st PCR ブランクを代わりに使用可能ですが、フィールドブランクとその他のブランクの両方を併せて利用することはできません。**ブランクの数は 10 以上必要**です。繰り返しますが、フィールドブランク、抽出ブランク、1st PCR ブランクの合計ではなく、いずれかが 10 以上です。

1st PCR 用のプライマーは、MiFish (Miya et al. 2015, 2020)、MiDeca (Komai et al. 2019)、MiMammal (Ushio et al. 2017)、MiBird (Ushio et al. 2018b)、Amph16S (Sakata et al. 2022)、MtInsects-16S (Takenaka et al. 2023) などが既に開発されており、対象とする生物群に応じて適宜選択できるようになります。新たに開発する場合は、対象とする生物群、遺伝子座を絞り込んだ上で公共のデータベース上から塩基配列を収集し、変異の多い領域を適度な長さで挟んでいる変異のほとんどない領域を探して設計することになります。また、1st PCR 用プライマーには、シーケンサの読み始めとなる部分に NNNNNN を付加することがよくあります。これは、Illumina 社製シーケンサでは読み始めの塩基多様度が低いと蛍光強度が飽和して正常に解読できなくなるためです。一部のプライマー合成業者では、NNNNNN のほとんどが TTTTTT になってしまうため、業者の選定に注意する必要があります。

2nd PCR 用のインデックスプライマーは、Illumina 社やサードパーティから既製品が販売されています。ま

た、筆者が開発したものを下記 URL にて公開しています。

<https://github.com/astanabe/TruSeqStyleIndexPrimers>

<https://github.com/astanabe/NexteraStyleIndexPrimers>

インデックス部分も塩基多様度が低いと正しく解読することができないため、使用するインデックスの組み合わせは慎重に検討する必要があります。どの位置でも AC と GT の比が 1:1 に近いことが望ましいとされています。特に、混合するサンプルが少ないときに注意が必要です。また、Claident でインデックスホッピングの検出・除去を行うには、各サンプルごとに「片方のインデックスを共有する、未使用のインデックスの組み合わせ」が 10 以上必要です。

内部標準 DNA 溶液は、合成業者から受け取った内部標準 DNA を TE バッファーなどで溶解し、蛍光色素を使用した濃度測定やデジタル PCR によって絶対定量して意図した濃度になるように希釈、混合したものを使用します。二本鎖 DNA 合成サービスとしては、ThermoFisher 社の Strings DNA Fragments や Integrated DNA Technologies 社の gBlocks といったものがあります。内部標準 DNA として使用する塩基配列は、使用するプライマーで解読できるインサート部分を公共のデータベースから収集し、変異が多い部分を GC 含量が変化しないようにしつつ無作為に 10% 以上変異させ、両端にプライマー配列を連結することで作成します。既知のどの生物からも 10% 以上、できれば 15% 以上異なるようになっていれば理想的です。MiFish プライマー用の内部標準 DNA 塩基配列であれば、Ushio et al. (2022) の Appendix S1 に掲載されています。

1.2.1 Claident における「サンプル ID」について

ここで、Claident の内部処理におけるサンプル ID について説明しておきます。通常、サンプル ID はユーザーが任意に指定すればいいわけですが、メタバーコーディングでは、同一のサンプルの同一のプライマー増幅産物を異なるシーケンスランで複数回シーケンスしたり、同一のサンプルの異なる複数のプライマーの増幅産物をシーケンスしたりすることがあるため、これらを識別するために Claident では以下の形式でサンプル ID を記述します。

`RunID__MaterialID__PrimerID`

RunID は、後述する解析コマンドの実行オプションとして指定する任意の文字列です。シーケンスラン (またはレーン) を識別するために使用されますので、ご自分でわかりやすいものにして下さい。PrimerID は、後述するファイルの中で指定する任意の文字列です。こちらは使用したプライマーを識別するために使用されます。MiFish プライマーを使用したのなら、MiFish でいいでしょう。MaterialID は、通常はサンプル ID として扱われる、サンプル物質に対してユーザーが割り当てた任意の文字列です。RunID や PrimerID は異なるが MaterialID が一致する場合、現物、すなわち鋳型 DNA は同一である、ということがわかります。つまり、現物サンプルと Claident でのサンプルは必ずしも 1 対 1 対応ではないため、上記のようなサンプル ID を使用することで対応する現物サンプルがサンプル ID のみでわかるように設計されています。

サンプルに反復を設けていることがあると思いますが、DNA 抽出・ライブラリ調製・シーケンスの全ての段階で区別している場合は別サンプルとして扱い、どこかの段階で区別しなく・できなくなるのであれば、同一のサンプルとして扱います。別サンプルとして扱う場合は、MaterialID の末尾に -R1 や -R2 などと付加することで、反復であることがわかるようにしておくのが良いでしょう。

なお、RunID・PrimerID・MaterialID には __ (2 個以上連続するアンダーバー) を含めることはできません。また、使用できる文字列は英数字とハイフンとアンダーバーのみです。その他の文字列が使用されていた場合、予期しないエラーが起きる可能性があります。

1.2.2 OTU と ASV について

Amplicon Sequence Variant (ASV) あるいは Exact Sequence Variant (ESV) は、「完全一致する配列、および完全一致すると推定された配列をまとめた分類単位」です。それに対して、Operational Taxonomic Unit (OTU: 操作的分類単位) は、その名の通り、「分析者が任意に設定した分類単位」です。なお、OTU は「塩基配列の類似度でクラスタリングした分類単位」であるという誤解がよくありますが、明らかに語義に反しているので注意して下さい。分析者が ASV を分類単位として解析する、と決めたのであれば、その ASV は OTU です。この後の記述や Claident の中では、OTU と ASV に区別はありません。

1.2.3 必要なファイル群とディレクトリ構造

ここでは、解析の前に用意する必要のあるファイル群を説明します。

1.2.3.1 ブランクリスト (blanklist.txt) 1 行に一つのブランクのサンプル ID を記述したテキストファイルです。以下のような形式で記述する必要があります。

```
RunID__BlankMaterialID1__PrimerID
RunID__BlankMaterialID2__PrimerID
RunID__BlankMaterialID3__PrimerID
```

Claident は、このファイルに記載されているものをブランクとして認識します。

1.2.3.2 濾過水量表 (watervoltable.tsv) 1 行に一つのサンプル ID とタブで区切って濾過水量の数値を記述したタブ区切りテキストファイルです。濾過フィルターが複数あって区別して記述したい場合、タブで区切って複数記述します (濃度推定時は合算して処理されます)。

```
RunID__SampleMaterialID1__PrimerID 1000 1000
RunID__SampleMaterialID2__PrimerID 1000 500
RunID__SampleMaterialID3__PrimerID 1500
RunID__BlankMaterialID1__PrimerID 500
RunID__BlankMaterialID2__PrimerID 500
RunID__BlankMaterialID3__PrimerID 500
```

この数値を使用して、元の環境水サンプル中における DNA 濃度が推定されます。単位は任意ですが、特段の理由がない限り mL で記述しておくのが良いでしょう。末尾にタブで区切って任意の文字列を付加することはできるので、単位を書いておくことも可能です。ただし、単位の異なる数値を換算して単位を統一するような処理には対応していません。

1.2.3.3 抽出 DNA 溶液量表 (solutionvoltable.tsv) 1 行に一つのサンプル・ブランク ID とタブで区切って抽出した DNA 溶液量の数値を記述したタブ区切りテキストファイルです。濾過フィルターが複数あり、抽出後の DNA 溶液も複数あって区別して記述したい場合、タブで区切って複数記述します (濃度推定時は合算して処理されます)。

```
RunID__SampleMaterialID1__PrimerID 200 200
RunID__SampleMaterialID2__PrimerID 200 200
RunID__SampleMaterialID3__PrimerID 200
RunID__BlankMaterialID1__PrimerID 200
RunID__BlankMaterialID2__PrimerID 200
RunID__BlankMaterialID3__PrimerID 200
```

この数値を使用して、抽出した DNA 溶液中の総 DNA コピー数が推定されます。単位は任意ですが、特段の理由がない限り μL で記述しておくのが良いでしょう。末尾にタブで区切って任意の文字列を付加することはできるので、単位を書いておくことも可能です。ただし、単位の異なる数値を換算して単位を統一するような処理には対応していません。

1.2.3.4 内部標準 DNA 塩基配列 (standard.fasta) FASTA 形式の内部標準 DNA 塩基配列ファイルです。複数の配列を記述することができます。以下は 4 つの内部標準 DNA 塩基配列を含む FASTA ファイルの例です。

```
>MiFish_STD_01
CACCGCGTTATACGACAGGCCCAAGTTGAACGCAGTCGGCGTAAAGAGTGGTTAAAAG...
>MiFish_STD_02
CACCGCGTTATACGACAGGCCCAAGTTGATCTTGAACGGCGTAAAGAGTGGTTAGATT...
>MiFish_STD_03
CACCGCGTTATACGACAGGCCCAAGTTGAAGCGACGGCGTAAAGAGTGGTTATCAC...
>MiFish_STD_04-2
CACCGCGTTATACGACAGGCCCAAGTTGAGATCCACGGCGTAAAGAGTGGTTAGAAC...
```

この塩基配列に基づいて内部標準 DNA が識別されます。塩基配列は、合成サービスに対して注文時に使用したものと同一、つまりプライマーのアニールする部位を含んでいても構いませんし、含んでいなくても構いません。

1.2.3.5 内部標準 DNA 濃度表 (stdconctable.tsv) サンプルごとに、1st PCR で添加した内部標準 DNA の濃度を記述したタブ区切りテキストファイルです。以下のような表形式にします。

samplename	MiFish_STD_01	MiFish_STD_02	MiFish_STD_03	MiFish_STD_04-2
RunID__SampleMaterialID1__PrimerID 5	10	20	40	
RunID__SampleMaterialID2__PrimerID 5	10	20	40	
RunID__SampleMaterialID3__PrimerID 5	10	20	40	
RunID__BlankMaterialID1__PrimerID 5	10	20	40	
RunID__BlankMaterialID2__PrimerID 5	10	20	40	
RunID__BlankMaterialID3__PrimerID 5	10	20	40	

濃度の単位は 1 μ L 当たりのコピー数です。ただし、これはサンプル DNA 溶液と等量の内部標準 DNA 溶液を添加して 1st PCR を行ったと仮定しています。したがって、サンプル DNA 溶液の 2 倍の内部標準 DNA 溶液を添加した場合は数値を 2 倍に、サンプル DNA 溶液を 10 倍希釈して希釈液と等量の内部標準 DNA 溶液を添加した場合は数値を 10 倍にします。

1.2.3.6 シーケンサの読み始めになる部分配列 (forwardprimer.fasta・reverseprimer.fasta) 1st PCR におけるフォワード側とリバース側のそれぞれのプライマー配列の一部を記述した FASTA 形式ファイルです。2nd PCR におけるインデックスプライマーがアニールする部位を取り除くことで、シーケンサの解読対象になる部分だけにします。つまり、1st PCR でフォワード側プライマーとして MiFish-U-F ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNNNGTCGGTAAACTCGTGCCAGC を使用した場合、NNNNNNGTCGGTAAACTCGTGCCAGC を塩基配列として記述します。いずれのファイルにも複数のプライマー配列を記述することができますが、フォワード側プライマー配列ファイルの 1 本目のプライマー配列はリバース側プライマー配列ファイルの 1 本目のプライマー配列とセットで検出されるため、リバース側プライマー配列ファイルの 2 本目以降のプライマー配列との組み合わせは検討されません。塩基配列には、R や Y や M や K や N などの、縮重塩基コードを使用可能です。MiFish のように僅かに異なる塩基配列のプライマーが提案されており、それらを複数混合して使用した場合、多重整列を行って縮重コンセンサス配列を記述します。例えば、MiFish-E-v2 と MiFish-U と MiFish-U2 を混合して使用した場合、フォワード側プライマー配列ファイル forwardprimer.fasta の内容は以下のようになります。

```
>MiFish
NNNNNNNGYYGGTAAAWCTCGTGCCAGC
```

上記の縮重コンセンサス配列の元になった配列は以下の通りです (見やすくするため整列してあります)。

```
>MiFish-E-F-v2
NNNNNNRGTGGTAAATCTCGTGCCAGC
>MiFish-U-F
NNNNNNGTCGGTAAACTCGTGCCAGC
>MiFish-U2-F
NNNNNNGCCGGTAAACTCGTGCCAGC
```

リバース側プライマー配列ファイル reverseprimer.fasta は以下のようになります。

```
>MiFish
NNNNNNNCATAGKRGGGTRTCTAATCCYMGTTTG
```

上記の縮重コンセンサス配列の元になった配列は以下の通りです (見やすくするため整列してあります)。

```
>MiFish-E-R-v2
NNNNNNGCATAGTGGGGTATCTAATCCTAGTTTG
>MiFish-U-R
NNNNNNCATAGTGGGGTATCTAATCCCAGTTTG
>MiFish-U2-R
NNNNNNCATAGGAGGGTGTCTAATCCCCGTTTG
```

これらのファイルの塩基配列名は、Claident のサンプル ID における PrimerID として使用されますので、上述のファイル群における PrimerID と一致している必要があります。

1.2.3.7 インデックスとして読まれる部分配列 (index1.fasta・index2.fasta) 2nd PCR におけるインデックスプライマーのインデックスとして解読される部分のみを取り出した FASTA 形式のファイルです。index2 (i5 index) はフォワード側インデックスプライマー内のインデックスで、解読の向きは機種によって異なります。index1 (i7 index) はリバース側インデックスプライマー内のインデックスで、発注時のプライマー配列とは逆向きに解読されます。Illumina 社シーケンサの SampleSheet.csv 内のインデックス配列は、解読方向が標準化されたものになっているので、これを取り出せば良いはずです。リバース側インデックス配列ファイル index1.fasta の内容は以下のようになります。

```
>SampleMaterialID1
ACCTGCAA
>SampleMaterialID2
GTTCTTG
>SampleMaterialID3
CCAGATCT
>BlankMaterialID1
AAGTGGA
>BlankMaterialID2
CCATGATC
>BlankMaterialID3
TCATGTCT
```

フォワード側インデックス配列ファイル index2.fasta も塩基配列が異なる以外は index1.fasta と内容は同じです。配列の名前が MaterialID と一致すること、配列の並び順が完全に同一であることが必要ですので注意して下さい。

1.2.3.8 undemultiplexed FASTQ 通常、受託解析業者に依頼すると SampleSheet.csv の内容に合わせてデマルチプレックス済みの FASTQ ファイルを納品されることが多いでしょう。しかし、Illumina 社製のデマルチプレックスプログラムはあまりに多くのサンプルを 1 シーケンスランや 1 レーンにデマルチプレックスすると正常にデマルチプレックスできなかったり、インデックスの塩基の信頼性を考慮していなかったり、1 塩基の読み間違い (不一致) を許容する設定であったり、「未使用のインデックスの組み合わせ」の塩基配列は全て破棄されてインデックスホッピングの検出に対応できなくなるため、Claident では内蔵するデマルチプレックスプログラム clsplitseq でのデマルチプレックスを推奨しています。

clsplitseq でのデマルチプレックスを行うには、Linux マシンに Illumina 社が提供する bcl2fastq というプログラムをインストールし、シーケンサのランデータからインデックス配列を含むデマルチプレックスしていない FASTQ (undemultiplexed FASTQ) を生成する必要があります。bcl2fastq は以下の URL から取得できます。

https://jp.support.illumina.com/sequencing/sequencing_software/bcl2fastq-conversion-software.html

執筆時点の最新版は v2.20 です。Debian・Ubuntu・Linux Mint の場合、Linux rpm と書かれている配布ファイルをダウンロードして作業ディレクトリに置き、ターミナルで以下のコマンドを実行することでインストール

できます。

```
sudo apt install rpm2cpio cpio
cd workingdirectory
unzip bcl2fastq2-v2-20-0-linux-x86-64.zip
mkdir temporary
cd temporary
rpm2cpio ../bcl2fastq2-v2.20.0.422-Linux-x86_64.rpm | cpio -id
sudo mkdir -p /usr/local/bin
sudo cp usr/local/bin/bcl2fastq /usr/local/bin/
sudo cp -R usr/local/share/css /usr/local/share/
sudo cp -R usr/local/share/xsl /usr/local/share/
cd ..
rm -rf temporary bcl2fastq2-v2.20.0.422-Linux-x86_64.rpm
```

なお、このプログラムは macOS には対応していません。macOS 上で実行するには、仮想マシンプログラムをインストールして仮想マシン上に Linux をインストールし、その Linux 上に bcl2fastq をインストールする必要があります。

bcl2fastq で undemultiplexed FASTQ を生成するには、SampleSheet.csv をコピーして Dummy.csv を作成し、テキストエディタで開いて [Data] セクションを編集します。[Data] セクションには 1 行目に各列のラベルが記されており、2 行目以降にサンプル名やインデックス配列が記されていますが、2 行目以降は削除します。FASTQ 生成の際にこのファイルをサンプルシートとして指定することで、bcl2fastq に内蔵されているデマルチプレックス機能を無効化し、undemultiplexed FASTQ を作成することができます。8 塩基長のデュアルインデックスでフォワード側 151 サイクル、リバース側 151 サイクル解読した場合、以下のコマンドで undemultiplexed FASTQ を 01_undemultiplexed ディレクトリに出力することができます。

```
bcl2fastq \
--processing-threads NumberOfCPUcores \
--create-fastq-for-index-reads \
--use-bases-mask Y150n,I8,I8,Y150n \
--runfolder-dir RunDataDirectory \
--sample-sheet Dummy.csv \
--output-dir 01_undemultiplexed
```

ここで、RunDataDirectory は、シーケンサ本体、またはシーケンサに付属の解析マシンに保存されている BaseCalls という名前のディレクトリを含むディレクトリです。予め bcl2fastq をインストールしたマシンにコピーしておく必要があります。NumberOfCPUcores は処理中に使用する CPU コア数の整数値で置き換えて下さい。

上記コマンドを実行すると、以下の 4 ファイルが生成されます。

～_I1_001.fastq.gz index1 の undemultiplexed FASTQ (長さ 8 塩基)
～_I2_001.fastq.gz index2 の undemultiplexed FASTQ (長さ 8 塩基)
～_R1_001.fastq.gz インサートのフォワード側リードの undemultiplexed FASTQ (長さ 150 塩基)
～_R2_001.fastq.gz インサートのリバース側リードの undemultiplexed FASTQ (長さ 150 塩基)

なお、NextSeq 1000・2000 や NovaSeq X などの新しい機種では、BCL Convert というまた別のプログラムを使用するように変更されています。

1.2.3.9 ディレクトリ構造 解析開始前の作業ディレクトリ内のファイルとディレクトリは以下の通りです。

- 作業ディレクトリ
 - blanklist.txt
 - watervoltable.tsv
 - solutionvoltable.tsv
 - standard.fasta
 - stdconctable.tsv
 - forwardprimer.fasta
 - reverseprimer.fasta
 - index1.fasta
 - index2.fasta
 - 01_undemultiplexed (ディレクトリ)
 - * ~_I1_001.fastq.gz
 - * ~_I2_001.fastq.gz
 - * ~_R1_001.fastq.gz
 - * ~_R2_001.fastq.gz

1.3 塩基配列データ処理

ここから実際の塩基配列データ処理の方法を説明していきます。全てのコマンドはターミナル上で実行します。作業ディレクトリがカレントディレクトリになっていると仮定しています。コマンドのオプションに含まれている NumberOfCPUcores は処理中に使用する CPU コア数の整数値で置き換えて下さい。これ以前に説明済みのファイルに関しては改めて説明しません。また、いくつかの処理ではディスクに激しくアクセスするため、低速なディスクに作業ディレクトリを設置していると大きく影響を受けます。作業ディレクトリは高速な SSD に設置することを強くお勧めします。

1.3.1 clsplitseq によるデマルチプレクシング

デマルチプレクシングを行うには、以下のコマンドを実行します。

```
clsplitseq \  
--runname=RunID \  
--forwardprimerfile=forwardprimer.fasta \  
--reverseprimerfile=reverseprimer.fasta \  
--truncateN=enable \  
--index1file=index1.fasta \  
--index2file=index2.fasta \  
--minqualtag=30 \  
--compress=xz \  
--seqnamestyle=illumina \  
--numthreads=NumberOfCPUcores \  

```

```
01_undemultiplexed/Undemultiplexed_R1_001.fastq.gz \
01_undemultiplexed/Undemultiplexed_I1_001.fastq.gz \
01_undemultiplexed/Undemultiplexed_I2_001.fastq.gz \
01_undemultiplexed/Undemultiplexed_R2_001.fastq.gz \
02_demultiplexed
```

それぞれのコマンドラインオプションの意味は以下の通りです。

```
--runname  任意の RunID を与える
--forwardprimerfile  フォワード側プライマー配列ファイル
--reverseprimerfile  リバース側プライマー配列ファイル
--truncateN  プライマー配列の一致度を算出する際に先頭の NNNNNN を除外するか否か
--index1file  リバース側インデックス配列ファイル
--index2file  フォワード側インデックス配列ファイル
--minqualtag  インデックス配列の品質値下限
--compress  圧縮形式の指定 (GZIP | BZIP2 | XZ | DISABLE から選択)
--seqnamestyle  塩基配列名の形式
```

コマンドラインオプション後に入力ファイル群、出力フォルダ名を与えます。なお、入力ファイルは以下の順で指定します。

1. インサートのフォワード側リードの undemultiplexed FASTQ
2. index1 の undemultiplexed FASTQ
3. index2 の undemultiplexed FASTQ
4. インサートのリバース側リードの undemultiplexed FASTQ

これは、Illumina 社シーケンサが解読する順になっています。

このコマンドでは、「未使用のインデックスの組み合わせ」を MaterialID とするサンプルの塩基配列も出力されます。後述するインデックスホッピングの検出・除去処理においてそれらのサンプルが使用されます。

データサイズが大きいと、この処理は非常に長い時間がかかります。

1.3.2 clconcatpairv によるペアエンド配列の連結

デマルチプレックスが終わったら、以下のコマンドでペアエンド配列を連結します。

```
clconcatpairv \
--mode=ovl \
--compress=xz \
--numthreads=NumberOfCPUcores \
02_demultiplexed \
03_concatenated
```

コマンドラインオプションの意味は以下の通りです。

```
--mode  Overlapped Paired-End か Non-overlapped Paired-End なのかを OVL または NON で指定
```

--compress 圧縮形式の指定 (GZIP | BZIP2 | XZ | DISABLE から選択)

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

1.3.3 clfilterseqv による低品質配列の除去

以下のコマンドで連結した配列に対して品質値から予想される期待エラー数を算出し、低品質の配列を除去します (Edgar and Flyvbjerg 2015)。

```
clfilterseqv \  
--maxqual=41 \  
--minlen=100 \  
--maxlen=250 \  
--maxnee=2.0 \  
--maxnNs=0 \  
--compress=xz \  
--numthreads=NumberOfCPUcores \  
03_concatenated \  
04_filtered
```

コマンドラインオプションの意味は以下の通りです。

--maxqual 品質値の上限 (超えた値はこの値になる)

--minlen 塩基配列長の下限

--maxlen 塩基配列長の上限

--maxnee 期待エラー数上限

--maxnNs 塩基配列中の N の数の上限

--compress 圧縮形式の指定 (GZIP | BZIP2 | XZ | DISABLE から選択)

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。ここで品質値の上限を指定しているのは、後述するデノイズングの際にあまりに品質値が大きい配列があるとエラーになることがあるためです。期待エラー数の多い配列や N を含む配列を除外しているのも同じ理由です。塩基配列長の上限下限は事前に予想されるインサート長に基づいて決定します。データから期待エラー数上限や塩基配列長の上限下限を決めたい場合、clcalcfastqstatv コマンドの出力を参考にすると良いかもしれません。

1.3.4 cldenoiseseqd によるデノイズング

以下のコマンドで DADA2 (Callahan et al. 2016) によるデノイズング処理を適用します。

```
cldenoiseseqd \  
--pool=pseudo \  
--numthreads=NumberOfCPUcores \  
04_filtered \  
05_denoised
```

コマンドラインオプションの意味は以下の通りです。

--pool サンプルのプール方法を指定 (ENABLE | DISABLE | PSEUDO から選択)

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

サンプルのプールを有効化すると、デノイジング効率は向上しますが、サンプル数が多いほど計算量が膨大になります。無効化すればデノイジング効率が低下してしまうため、DADA2 開発者が用意している Pseudo-pooling 法をここでは使用しています。Pseudo-pooling 法に関しては DADA2 の公式 Web サイトをご参照下さい。

1.3.5 clremovechimev による参照配列データベースを用いないキメラ除去

以下のコマンドで VSEARCH (Rognes et al. 2016) に実装されている UCHIME3 アルゴリズム (Edgar 2016) を使用したキメラ配列検出・除去を適用します。

```
clremovechimev \  
--mode=denovo \  
--uchimedenovo=3 \  
--numthreads=NumberOfCPUcores \  
05_denoised \  
06_chimeraremoved
```

コマンドラインオプションの意味は以下の通りです。

--mode 動作モードを指定 (BOTH | DENOVO | REF から選択)

--uchimedenovo UCHIME de novo のバージョンを指定 (1 | 2 | 3 から選択)

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

--mode=denovo というのは参照配列データベースを用いないキメラ除去モードのことを指します。UCHIME de novo は多少内容の異なる 3 つのバージョンがありますが、デノイジングした塩基配列に対して最適化されているのは UCHIME3 なので、それを選択しています。

1.3.6 clclusterstdv による内部標準配列クラスタリング

以下のコマンドで VSEARCH (Rognes et al. 2016) に実装されている UCLUST アルゴリズム (Edgar 2010) を使用して内部標準配列にマッチする塩基配列をひとまとめにします。

```
clclusterstdv \  
--standardseq=standard.fasta \  
--minident=0.9 \  
--numthreads=NumberOfCPUcores \  
06_chimeraremoved \  
07_stdclustered
```

コマンドラインオプションの意味は以下の通りです。

--standardseq 内部標準 DNA 塩基配列ファイル

--minident 内部標準 DNA と判定する類似度の下限

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

内部標準 DNA と判定する類似度の下限は、内部標準配列と実在する生物の塩基配列の類似度最大値が低い

(0.85 未満) 場合には 0.9 程度で問題ないでしょう。Ushio et al. (2022) の Appendix S1 に掲載されている MiFish 用内部標準配列はこの条件を満たしています。内部標準配列と実在する生物の塩基配列の類似度が高く (0.85 以上)、内部標準 DNA の合成エラー率が低いと期待できる場合は 0.97 程度まで値を大きくしても構いません。内部標準 DNA の合成エラー率が低いと期待できるかどうかは、合成業者の公称エラー率や合成方法などから判断します。判断が難しい場合は、値を 0.90~0.97 まで 0.01 間隔で変化させ、内部標準 DNA と判定される配列数が急激に変化するところを探し、変化点の小さい方に設定します。内部標準 DNA と判定される配列数が急激に変化するところが見つからない場合、内部標準 DNA の合成エラー率が非常に高く定量は不可能と考えられるため、内部標準 DNA の合成を業者に依頼するところから全てやり直す必要があります。

1.3.7 clremovechimev による参照配列データベースを用いたキメラ除去

以下のコマンドで VSEARCH (Rognes et al. 2016) に実装されている UCHIME アルゴリズム (Edgar et al. 2011) を使用したキメラ配列検出・除去を適用します。

```
clremovechimev \  
--mode=ref \  
--referencedb=cdu12s \  
--addtoref=07_stdclustered/stdvariations.fasta \  
--numthreads=NumberOfCPUcores \  
07_stdclustered \  
08_chimeraremoved
```

コマンドラインオプションの意味は以下の通りです。

--mode 動作モードを指定 (BOTH | DENOVO | REF から選択)
--referencedb 参照配列データベース
--addtoref 参照配列データベースに追加する参照配列ファイル

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

--mode=ref は参照配列データベースを用いたキメラ除去モードを指します。Claident のインストーラで自動インストールされる参照配列データベースは以下の通りです。

rdpgoldv9 細菌 16S 用
dairydb3.0.0 細菌 16S 用
unite20170628, unite20170628untrim, unite20170628its1, unite20170628its2 真菌 ITS 用
cdu12s ミトコンドリア 12S 用
cdu16s ミトコンドリア 16S 用
cducox1 ミトコンドリア COX1(COI) 用
cducytb ミトコンドリア Cyt-b 用
cdudloop ミトコンドリア D-loop(調節領域) 用
cdumatk 葉緑体 matK 用
cdurbcl 葉緑体 rbcL 用
cdutrnhpsba 葉緑体 trnH-psbA 用

手動でインストールする必要がありますが、細菌 16S には SILVA の SSURef や SSUParc、真菌 ITS には UNITE の Full UNITE+INSd dataset for eukaryotes を推奨します。MiFish で増幅されるのはミトコンドリア 12S 領域の一部なので、cdu12s を使用します。名前が cdu から始まるキメラ検出用参照配列データベースは、筆者が公共データベースの完全長または完全長に近い長さのミトコンドリアゲノム・葉緑体ゲノム配列から当該領域を切り出したものです。完全長または完全長に近いデータはキメラである可能性は低いだろうという仮定に基づいています。内部標準 DNA を添加して行う PCR では、内部標準 DNA と内部標準 DNA 間のキメラや、内部標準 DNA と生物の DNA 間のキメラも形成されます。そこで、内部標準 DNA と判定された配列群 (07_stdclustered/stdvariations.fasta に含まれている) を参照配列に追加することで、キメラの検出力向上を狙っています。standard.fasta (合成業者に依頼した際の配列、すなわち合成エラーを一切含まない配列) ではなく 07_stdclustered/stdvariations.fasta (不一致をある程度許容して内部標準配列と判定された配列、すなわち合成エラーを含む内部標準配列) を使用するの、合成エラーのある内部標準 DNA と合成エラーのある内部標準 DNA 間のキメラや合成エラーのある内部標準 DNA と生物の DNA 間のキメラをできるだけ検出するためです。

1.3.8 clremovecontam によるインデックスホッピング除去

以下のコマンドで、Esling et al. (2015) の方法に基づくインデックスホッピング除去を適用します。

```
clremovecontam \
--test=thompson \
--index1file=index1.fasta \
--index2file=index2.fasta \
--numthreads=NumberOfCPUcores \
08_chimeraremoved \
09_hoppingremoved
```

コマンドラインオプションの意味は以下の通りです。

--test 検定方法を指定 (THOMPSON | BINOMIAL から選択)

--index1file リバース側インデックス配列ファイル (clsplitseq に与えたものと同じ)

--index2file フォワード側インデックス配列ファイル (clsplitseq に与えたものと同じ)

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

このコマンドは、各サンプルに対して、「片方のインデックスを共有する、未使用のインデックスの組み合わせ」(共有していない方のインデックスのインデックスホッピングによって生じたものである可能性がある) におけるその ASV のリード数に対して、サンプルにおける ASV のリード数が外れ値でないのであれば、それはインデックスホッピング由来であると判定して 0 に置換します。

1.3.9 clremovecontam とネガティブコントロールを利用したデコンタミネーション

以下のコマンドでは、サンプルとフィールドブランクにおける環境水中の各 ASV の DNA 濃度を算出し、サンプルにおける DNA 濃度が外れ値でないならば、それはコンタミネーション由来であると判定して 0 に置換します。


```

clremovecontam \
--test=thompson \
--blanklist=blanklist.txt \
--stdconctable=stdconctable.tsv \
--solutionvtable=solutionvtable.tsv \
--watervtable=watervtable.tsv \
--numthreads=NumberOfCPUcores \
09_hoppingremoved \
10_decontaminated

```

コマンドラインオプションの意味は以下の通りです。

--test 検定方法を指定 (THOMPSON | BINOMIAL から選択)
--blanklist ブランクのサンプル ID リスト
--stdconctable 内部標準 DNA 濃度表のタブ区切りテキスト
--solutionvtable 抽出 DNA 溶液量表のタブ区切りテキスト
--watervtable 濾過水量表のタブ区切りテキスト

コマンドラインオプションに引き続いて、入力フォルダ、出力フォルダを指定します。

なお、抽出 DNA 溶液量表と濾過水量表がなく、内部標準 DNA 濃度表のみが与えられた場合、環境水中の DNA 濃度の代わりに抽出 DNA 溶液中の DNA 濃度を算出し、その値に基づいてデコンタミネーションを行います。抽出 DNA 溶液量表も濾過水量表も内部標準 DNA 濃度表もない場合、リード数の値をそのまま使用してデコンタミネーションを行います。内部標準 DNA 濃度を使用した濃度推定値を使用する場合、ライブラリ調製において濃度均一化処理などを行っていても適用可能ですが、リード数の値をそのまま使用する場合、1) ライブラリ調製の過程で濃度均一化処理を一切行っていない、2) PCR の合計サイクル数は最小限に留めている (どのサンプルもプラトーに達していない)、必要があります。

塩基配列データ処理はここまでとなりますが、ここまでで得られた ASV をさらにクラスタリングしてまとめたい場合があると思います。そのような場合は、`clclassseqv` コマンドで追加のクラスタリングを行うことができます。

デノイジング以降、以下のようなファイルが出力フォルダには作成されています (ただし〜は 3 ファイルで共通)。

～.fasta この時点での ASV・OTU の塩基配列ファイル
 ～.otu.gz この時点での ASV・OTU の所属を記録したファイル
 ～.tsv この時点での ASV・OTU の各サンプルでのリード数表のタブ区切りテキスト

上記タブ区切りテキストの内容を追跡することで、各処理によって起きた変化がわかります。

1.4 分子同定

ここでは、QCauto 法と 95%-3NN 法 (Tanabe and Toju 2013) に基づく分子同定の手順を示します。QCauto 法は誤同定の非常に少ない方法ですが、その代わり種や属などの低レベル分類階層が「unidentified」になりやすい性質があります。95%-3NN 法は種や属などの低レベル分類階層まで同定できることが多いですが、参照配

列データベースの整備状況次第では誤同定が多くなってしまう性質があります。MiFish によるメタバーコーディングを日本の淡水域や日本近海のサンプルで行う場合、千葉県立博物館のグループによって参照配列データベースがよく整備されているため、95%-3NN 法でもそれほど問題は生じません。しかし、それ以外の参照配列データベースの網羅度が十分でない状況では、QCAuto 法の結果を使用することを推奨します。

この先に進む前に、以下のコマンドで作業ディレクトリに分子同定の出力ディレクトリを作成しておきます。

```
mkdir 11_taxonomy
```

1.4.1 分子同定用参照配列データベース

Claident では、標準で多数の分子同定用参照配列データベースが添付されています。Claident に添付されているデータベースは、以下の形式で命名されています。

分類群_遺伝子座_参照配列同定情報の分類階層

分類群_遺伝子座 には以下のものがあります。

overall 全生物全遺伝子座
animals_COX1 動物 COX1(COI)
animals_mt 動物ミトコンドリアゲノム
eukaryota_LSU 真核生物 LSU(28S)
eukaryota_SSU 真核生物 SSU(18S)
fungi_all 真菌全遺伝子座
fungi ITS 真菌 ITS
plants_cp 植物葉緑体ゲノム
plants_matK 植物 matK
plants_rbcL 植物 rbcL
plants_trnH-psbA 植物 trnH-psbA
prokaryota_16S 原核生物 16S
prokaryota_all 原核生物全遺伝子座

参照配列同定情報の分類階層 には以下のものがあります。

class 綱以下の同定情報のある参照配列を含む (overall のみ)
order 目以下の同定情報のある参照配列を含む (overall のみ)
family 科以下の同定情報のある参照配列を含む (overall のみ)
genus 属以下の同定情報のある参照配列を含む
species_wsp 種以下の同定情報がある参照配列を含む。種名に「sp.」が含まれる参照配列は除外されていない
species 種以下の同定情報がある参照配列を含むが、種名の末尾に「sp.」が含まれる参照配列は除外されている
species_wosp 種以下の同定情報がある参照配列を含むが、種名に「sp.」が含まれる参照配列は除外されている

genus_man 属以下の同定情報があり、属名が空欄でない参照配列を含む

species_wsp_man 種以下の同定情報がある参照配列を含む。種名に「sp.」が含まれる参照配列は除外されていないが、属名が空欄の参照配列は除外されている

species_man 種以下の同定情報がある参照配列を含むが、種名の末尾に「sp.」が含まれる、または属名が空欄の参照配列は除外されている

species_wosp_man 種以下の同定情報がある参照配列を含むが、種名に「sp.」が含まれる、または属名が空欄の参照配列は除外されている

データベースの種類が多すぎて使い分けが難しいのですが、どれが最適なのかは分類群や研究目的によって異なります。MiFish によるメタバーコーディングを日本の淡水域や日本近海のサンプルで行う場合、動物以外の配列やミトコンドリアゲノム以外の配列も同定したいなら、overall_species_wsp を推奨します。しかし、overall 系データベースは巨大で、搭載しているメモリが少ないマシンではメモリ不足になってしまいます。そのような場合、動物以外の配列やミトコンドリアゲノム以外の配列は同定できなくなりますが、animals_mt_species_wsp が良いでしょう。真菌や細菌などで属レベルの同定が非常に重要なケースでは、~_species_wsp_man を使うと良いかもしれません。使い分けに悩んだ場合は、各データベースを使用して同定した結果をマージしていいとこ取りすることができますので、全部やってみていただければいいでしょう。

1.4.2 clmakecachedb によるキャッシュデータベースの生成

最初に、以下のコマンドで分子同定に用いるキャッシュデータベースの生成を行います。

```
clmakecachedb \  
--blastdb=animals_mt_species_wsp \  
--ignoreotuseq=standard.fasta \  
--numthreads=NumberOfCPUcores \  
10_decontaminated/decontaminated.fasta \  
11_taxonomy/cachedb_species_wsp
```

コマンドラインオプションの意味は以下の通りです。

--blastdb 使用する分子同定用参照配列データベース

--ignoreotuseq 指定した FASTA 配列ファイルに含まれる配列名と一致する OTU は無視する

コマンドラインオプションに引き続いて、入力ファイル、出力フォルダを指定します。

大量のメモリを使用する可能性があるため、実行中はもう一つターミナルを起動して空きメモリ量を top コマンドなどを実行して監視し、もし空きメモリがなくなるようであれば Ctrl+C キーを押して強制終了して使用するデータベースを変更したりマシンにメモリを増設することを検討して下さい。

1.4.3 QCauto 法による分子同定

1.4.3.1 clidentseq による近隣配列群の取得 ほげほげ

```
clidentseq \  
--method=QC \  
--blastdb=11_taxonomy/cachedb_species_wsp \  
--ignoreotuseq=standard.fasta \  
--numthreads=NumberOfCPUcores \  
10_decontaminated/decontaminated.fasta \  
11_taxonomy/neighborhoods_qc_species_wsp.txt
```

ほげほげ

1.4.3.2 classigntax による分類群の割当 ほげほげ

```
classigntax \  
--taxdb=animals_mt_species_wsp \  
11_taxonomy/neighborhoods_qc_species_wsp.txt \  
11_taxonomy/taxonomy_qc_species_wsp.tsv
```

ほげほげ

1.4.4 95%-3NN 法による分子同定

1.4.4.1 clidentseq による近隣配列群の取得 ほげほげ

```
clidentseq \  
--method=3,95% \  
--blastdb=11_taxonomy/cachedb_species_wsp \  
--ignoreotuseq=standard.fasta \  
--numthreads=NumberOfCPUcores \  
10_decontaminated/decontaminated.fasta \  
11_taxonomy/neighborhoods_3nn_species_wsp.txt
```

ほげほげ

1.4.4.2 classigntax による分類群の割当 ほげほげ

```
classigntax \  
--taxdb=animals_mt_species_wsp \  
--minnsupporter=1 \  
11_taxonomy/neighborhoods_3nn_species_wsp.txt \  
11_taxonomy/taxonomy_3nn_species_wsp.tsv
```

ほげほげ

1.4.5 clmakeidentdb による分子同定結果の再利用

ほげほげ

```
clmakeidentdb \  
--append \  
11_taxonomy/neighborhoods_qc_species_wsp.txt \  
11_taxonomy/qc_species_wsp.identdb
```

```
clmakeidentdb \  
--append \  
11_taxonomy/neighborhoods_3nn_species_wsp.txt \  
11_taxonomy/3nn_species_wsp.identdb
```

clmakecachedb と clidentseq

1.4.6 clmergeassign による複数の分子同定結果のマージ

ほげほげ

```
clmergeassign \  
--preferlower \  
--priority=descend \  
11_taxonomy/taxonomy_qc_species_wsp.tsv \  
11_taxonomy/taxonomy_3nn_species_wsp.tsv \  
11_taxonomy/taxonomy_merged.tsv
```

ほげほげ

1.4.7 clfillassign による分子同定結果の穴埋め

ほげほげ

```
clfillassign \  
11_taxonomy/taxonomy_merged.tsv \  
11_taxonomy/taxonomy_merged_filled.tsv
```

ほげほげ

1.5 サンプル × OTU 表の作成

ほげほげ

```
mkdir -p 12_community
```

```
cp \  
10_decontaminated/decontaminated.tsv \  
12_community/sample_otu_matrix_raw.tsv
```

1.5.1 clfiltersum によるサンプル × OTU 表の加工

ほげほげ

```
clfiltersum \  
--otuseq=standard.fasta \  
12_community/sample_otu_matrix_raw.tsv \  
12_community/sample_otu_matrix_standard.tsv
```

```
clfiltersum \  
--taxfile=11_taxonomy/taxonomy_merged_filled.tsv \  
--includetaxa=class,Hyperoartia,class,Myxini,class,Chondrichthyes \  
--includetaxa=superclass,Actinopterygii,order,Coelacanthiformes \  
--includetaxa=subclass,Dipnomorpha \  
10_decontaminated/decontaminated.tsv \  
12_community/sample_otu_matrix_fishes.tsv
```

ほげほげ

1.5.2 clrarefysum によるサンプル × OTU 表のカバレッジベースレアファクション

ほげほげ

```
clrarefysum \  
--minpcov=0.99 \  
--minnread=1000 \  
--nreps=10 \  
--numthreads=NumberOfCPUcores \  
12_community/sample_otu_matrix_fishes.tsv \  
12_community/sample_otu_matrix_fishes_rarefied
```

ほげほげ

1.5.3 clestimateconc と内部標準 DNA リード数を用いた DNA 濃度の推定

ほげほげ

```
clestimateconc \  
--stdconctable=stdconctable.tsv \  
--stdtable=12_community/sample_otu_matrix_standard.tsv \  
--solutionvoltage=solutionvoltage.tsv \  
--watervoltage=watervoltage.tsv \  
--numthreads=NumberOfCPUcores \  
12_community/sample_otu_matrix_fishes_rarefied01.tsv \  
12_community/sample_otu_matrix_fishes_rarefied01_estimated.tsv
```

ほげほげ

1.5.4 サンプル × OTU 表を用いた群集生態学的解析

セルが自然数であることを仮定→非定量時系列データ分析など、定量データである必要がある場合→定量サンプル間・異なるシーケンスランのサンプル間での比較をしたい場合→定量シングルトンやダブルトンの数から近似的にカバレッジを推定する手法は使用できない。Chao 指数はダメ iNEXT はダメ

引用文献

- Callahan, B. J., P. J. McMurdie, M. J. Rosen, A. W. Han, A. J. A. Johnson, and S. P. Holmes. 2016. DADA2: High-resolution sample inference from Illumina amplicon data. *Nature Methods* 13:581–583.
- Chao, A., and L. Jost. 2012. Coverage-based rarefaction and extrapolation: Standardizing samples by completeness rather than size. *Ecology* 93:2533–2547.
- Edgar, R. C. 2010. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* 26:2460–2461.
- Edgar, R. C. 2016, September. UCHIME2: Improved chimera prediction for amplicon sequencing. *bioRxiv*.
- Edgar, R. C., and H. Flyvbjerg. 2015. Error filtering, pair assembly and error correction for next-generation sequencing reads. *Bioinformatics* 31:3476–3482.
- Edgar, R. C., B. J. Haas, J. C. Clemente, C. Quince, and R. Knight. 2011. UCHIME improves sensitivity and speed of chimera detection. *Bioinformatics* 27:2194–2200.
- Esling, P., F. Lejzerowicz, and J. Pawlowski. 2015. Accurate multiplexing and filtering for high-throughput amplicon-sequencing. *Nucleic Acids Research* 43:2513–2524.
- Komai, T., R. O. Gotoh, T. Sado, and M. Miya. 2019. Development of a new set of PCR primers for eDNA metabarcoding decapod crustaceans. *Metabarcoding and Metagenomics* 3:e33835.
- Miya, M., R. O. Gotoh, and T. Sado. 2020. MiFish metabarcoding: A high-throughput approach for simultaneous detection of multiple fish species from environmental DNA and other samples. *Fisheries Science* 86:939–970.
- Miya, M., Y. Sato, T. Fukunaga, T. Sado, J. Y. Poulsen, K. Sato, T. Minamoto, S. Yamamoto, H. Yamanaka, H. Araki, M. Kondoh, and W. Iwasaki. 2015. MiFish, a set of universal PCR primers for metabarcoding environmental DNA from fishes: Detection of more than 230 subtropical marine species. *Royal Society Open Science* 2:150088.
- Rognes, T., T. Flouri, B. Nichols, C. Quince, and F. Mahé. 2016. VSEARCH: A versatile open source tool for metagenomics. *PeerJ* 4:e2584.
- Sakata, M. K., M. U. Kawata, A. Kurabayashi, T. Kurita, M. Nakamura, T. Shirako, R. Kakehashi, K. Nishikawa, M. Y. Hossman, T. Nishijima, J. Kabamoto, M. Miya, and T. Minamoto. 2022. Development and evaluation of

- PCR primers for environmental DNA (eDNA) metabarcoding of Amphibia. *Metabarcoding and Metagenomics* 6:e76534.
- Sato, Y., M. Miya, T. Fukunaga, T. Sado, and W. Iwasaki. 2018. MitoFish and MiFish Pipeline: A Mitochondrial Genome Database of Fish with an Analysis Pipeline for Environmental DNA Metabarcoding. *Molecular Biology and Evolution* 35:1553–1555.
- Takenaka, M., K. Yano, T. Suzuki, and K. Tojo. 2023. Development of novel PCR primer sets for DNA barcoding of aquatic insects, and the discovery of some cryptic species. *Limnology* 24:121–136.
- Tanabe, A. S., and H. Toju. 2013. Two New Computational Methods for Universal DNA Barcoding: A Benchmark Using Barcode Sequences of Bacteria, Archaea, Animals, Fungi, and Land Plants. *PLOS ONE* 8:e76910.
- Ushio, M., H. Fukuda, T. Inoue, K. Makoto, O. Kishida, K. Sato, K. Murata, M. Nikaido, T. Sado, Y. Sato, M. Takeshita, W. Iwasaki, H. Yamanaka, M. Kondoh, and M. Miya. 2017. Environmental DNA enables detection of terrestrial mammals from forest pond water. *Molecular Ecology Resources* 17:e63–e75.
- Ushio, M., S. Furukawa, H. Murakami, R. Masuda, and A. J. Nagano. 2022. An efficient early-pooling protocol for environmental DNA metabarcoding. *Environmental DNA* 4:1212–1228.
- Ushio, M., H. Murakami, R. Masuda, T. Sado, M. Miya, S. Sakurai, H. Yamanaka, T. Minamoto, and M. Kondoh. 2018a. Quantitative monitoring of multispecies fish environmental DNA using high-throughput sequencing. *Metabarcoding and Metagenomics* 2:e23297.
- Ushio, M., K. Murata, T. Sado, I. Nishiumi, M. Takeshita, W. Iwasaki, and M. Miya. 2018b. Demonstration of the potential of environmental DNA as a tool for the detection of avian species. *Scientific Reports* 8:4493.
- Zhu, T., Y. Sato, T. Sado, M. Miya, and W. Iwasaki. 2023. MitoFish, MitoAnnotator, and MiFish Pipeline: Updates in 10 Years. *Molecular Biology and Evolution* 40:msad035.