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)

2024-01-26

# 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。 Claidentの読みは「クライデン」です(末尾のtは発音しません)。 名前の由来は「CLAssification」と「IDENTification」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., 2020, 2015) 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
* 非定量および定量メタバーコーディング (Ushio, Murakami, et al., 2018) をサポート
* より柔軟で詳細な解析に対応
* Webサービスはなく、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。 本章のサポートページを下記URLに設置していますので、適宜ご参照下さい。

* <https://github.com/astanabe/eDNAmanual>

サンプルデータ、サンプルファイル、本章の原稿ファイル等が置いてあります。

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。

## 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)。 メモリも搭載しているうちの半分しか使用できません。 大きなデータ解析にはディスクやメモリの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。 ディスクアクセス速度がボトルネックになることも多いため、高速なSSDを搭載したマシンを用意して下さい。

Debian・Ubuntu・Linux Mintおよび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/installDB\_Debian.sh  
sh installClaident\_Debian.sh  
sh installOptions\_Debian.sh  
sh installUCHIMEDB\_Debian.sh  
sh installDB\_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/installDB\_macOSHomebrew.sh  
sh installClaident\_macOSHomebrew.sh  
sh installOptions\_macOSHomebrew.sh  
sh installUCHIMEDB\_macOSHomebrew.sh  
sh installDB\_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、実行コマンドが存在する「インストール先/bin」が環境変数PATHに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での解析を行う際には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での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れば、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。

## 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. OTU組成表の作成・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, Murakami, et al., 2018)

最終的に得られたOTU組成表を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、最後の溶出時に使用した液量)
* 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., 2020, 2015) 、 MiDeca (Komai et al., 2019) 、MiMammal (Ushio et al., 2017) 、MiBird (Ushio, Murata, et al., 2018) 、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つつあります。 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Nを6個程度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、NのほとんどがT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。

### 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複数の異なるシーケンスランでシーケンスしたり、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同一のシーケンスランで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。

### 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。 Claidentの中ではほとんどの場合OTUはASVになりますが、この先、OTUがASVでない可能性がある処理ではOTUと記述することがあります。

### 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。 ファイル名は任意ですが、後述するコマンドの中で仮定しているファイル名を記してあります。

#### 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。

#### 濾過水量表(watervoltable.tsv)

1行に一つのサンプルIDとタブ文字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、タブ文字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。 末尾にタブ文字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。

#### 抽出DNA溶液量表(solutionvoltable.tsv)

1行に一つのサンプル・ブランクIDとタブ文字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、タブ文字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。 末尾にタブ文字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。

なお、「DNA抽出で回収できた液量」ではなく、「DNA溶出に使用した液量」であることに注意して下さい。 つまり、スピンカラムや磁気ビーズに200 μLの溶出バッファーを添加して、最終的に得られたDNA抽出液が190 μLだった場合、回収できなかっただけで実際にはさらに10 μLのDNA抽出液が存在するので、ここには200 μLと書くのが正しいことになります。 ただし、DNAの溶出前に使用したウォッシュバッファーは完全に除去したと仮定しています。

#### 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  
CACCGCGGTTATACGACAGGCCCAAGTTGAACGCAGTCGGCGTAAAGAGTGGTTAAAAG...  
>MiFish\_STD\_02  
CACCGCGGTTATACGACAGGCCCAAGTTGATCTTGAACGGCGTAAAGAGTGGTTAGATT...  
>MiFish\_STD\_03  
CACCGCGGTTATACGACAGGCCCAAGTTGAAGCGACGCGGCGTAAAGAGTGGTTATCAC...  
>MiFish\_STD\_04-2  
CACCGCGGTTATACGACAGGCCCAAGTTGAGATCCCACGGCGTAAAGAGTGGTTAGAAC...

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(つまり、インサートであっても)構いません。 内部標準DNAの配列名は、後述する内部標準DNA濃度表と一致している必要があります。

#### 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 μL 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。 内部標準DNAの名前は、前述の内部標準DNA塩基配列の名前と一致している必要があります。

#### 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 ACACTCTTTCCCTACACGACGCTCTTCCGATCTNNNNNNGTCGGTAAAACTCGTGCCAGCを使用した場合、NNNNNNGTCGGTAAAACTCGTGCCAGC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  
NNNNNNRGTTGGTAAATCTCGTGCCAGC  
>MiFish-U-F  
 NNNNNNGTCGGTAAAACTCGTGCCAGC  
>MiFish-U2-F  
 NNNNNNGCCGGTAAAACTCGTGCCAGC

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。

#### 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社シーケンサで使用される「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  
GTTCCTTG  
>SampleMaterialID3  
CCAGATCT  
>BlankMaterialID1  
AAGTGTGA  
>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。

#### 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BCL Convert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。 以下の作業における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。

BCL Convertは下記URLからダウンロードできます。

* <https://jp.support.illumina.com/sequencing/sequencing_software/bcl-convert.html>

執筆時点での最新版はv4.2.4です。 Debian・Ubuntu・Linux MintおよびWindows上にインストールしたUbuntuの場合、(Oracle 8)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「workingdirectory」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 pstack  
cd workingdirectory  
mkdir temporary  
cd temporary  
rpm2cpio ../bcl-convert-4.2.4-2.el8.x86\_64.rpm | cpio -id  
sudo mkdir -p /usr/local/bin  
sudo cp usr/bin/bcl-convert /usr/local/bin/  
sudo mkdir -p /var/log/bcl-convert  
sudo chmod 777 /var/log/bcl-convert  
cd ..  
rm -rf temporary bcl-convert-4.2.4-2.el8.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。

BCL Convertでundemultiplexed FASTQを生成するには、下記の内容の「Dummy.csv」をテキストエディタで作成します。

[Header]  
FileFormatVersion,2  
[BCLConvert\_Settings]  
OverrideCycles,Y150N1;I8;I8;Y150N1  
CreateFastqForIndexReads,1  
[BCLConvert\_Data]  
Lane,Sample\_ID,index,index2  
1,Dummy,CCCCCCCC,CCCCCCCC

ターミナルで下記のコマンドを実行すれば、テキストエディタがなくても「Dummy.csv」を作成できます。

echo '[Header]  
FileFormatVersion,2  
[BCLConvert\_Settings]  
OverrideCycles,Y150N1;I8;I8;Y150N1  
CreateFastqForIndexReads,1  
[BCLConvert\_Data]  
Lane,Sample\_ID,index,index2  
1,Dummy,CCCCCCCC,CCCCCCCC' > Dummy.csv

これは、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(末尾1塩基破棄)、リバース側151サイクル(末尾1塩基破棄)の場合の設定ファイルですので、インデックス長やサイクル数が異なる場合は適宜変更する必要があります。 ダミーのサンプルとして、index1およびindex2が共にCCCCCCCCのDummyというサンプルの行があります。 サンプル行が一つもないとエラーになるため、このような行を作成する必要があります。 万が一、index1およびindex2が共にCCCCCCCCのサンプル、index1がCCCCCCCCのサンプル、index2がCCCCCCCCのサンプルの**いずれか**が実在する場合は適当な配列に書き換えて下さい。 index1およびindex2が共にCCCCCCCCのサンプルが実在しないのにサンプルDummyのFASTQにいくらかデータが出力されることがありますが、index1およびindex2が共にCCCCCCCCのサンプルが本当に実在しないのなら、シーケンスエラーによるものなので問題はありません。 なお、インデックスの長さが8塩基ではない場合は、OverrideCyclesの行とCCCCCCCCは書き換える必要があります。 例えば10塩基長のデュアルインデックスでフォワード側301サイクル(末尾1塩基破棄)、リバース側301サイクル(末尾1塩基破棄)の場合、OverrideCycles,Y300N1;I10;I10;Y300N1およびCCCCCCCCCCとします。

FASTQ生成の際にこのファイルを--sample-sheetに指定することで、BCL Convert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。 以下のコマンドでは、undemultiplexed FASTQを「01\_undemultiplexed」ディレクトリに出力することができます。

bcl-convert \  
--sample-sheet Dummy.csv \  
--bcl-input-directory RunDataDirectory \  
--output-directory 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、シーケンスランのデータが保存されているディレクトリです。 通常は「Data」というディレクトリが含まれているはずです。 このディレクトリを予めBCL Convert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。 なお、使用するCPU数はデフォルトで自動的に決定されます(搭載されている全CPUを使用します)。

上記の例ではレーンが一つしかない機種を想定しています。 レーンが複数ある機種のデータを扱う場合、--bcl-only-laneオプションを使用することで、特定のレーンのみのデータからundemultiplexed FASTQを生成できます。 1番目のレーンのデータだけをundemultiplexed FASTQにする場合、--bcl-only-lane 1とします。 このオプションを指定しない場合は全レーンのデータがレーンごとに異なるファイルに出力されます。

上述の通りにBCL Convertを実行すると、サンプルDummyのファイル以外に以下の4ファイルが生成されます(1レーンのみ出力した場合)。

Undetermined\_S0\_L001\_I1\_001.fastq.gz

index1のundemultiplexed FASTQ (長さ8塩基)

Undetermined\_S0\_L001\_I2\_001.fastq.gz

index2のundemultiplexed FASTQ (長さ8塩基)

Undetermined\_S0\_L001\_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)

Undetermined\_S0\_L001\_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)

#### ディレクトリ構造

Claidentでの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)
    - Undetermined\_S0\_L001\_I1\_001.fastq.gz
    - Undetermined\_S0\_L001\_I2\_001.fastq.gz
    - Undetermined\_S0\_L001\_R1\_001.fastq.gz
    - Undetermined\_S0\_L001\_R2\_001.fastq.gz

## 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。 そのため、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。

### 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/Undetermined\_S0\_L001\_R1\_001.fastq.gz \  
01\_undemultiplexed/Undetermined\_S0\_L001\_I1\_001.fastq.gz \  
01\_undemultiplexed/Undetermined\_S0\_L001\_I2\_001.fastq.gz \  
01\_undemultiplexed/Undetermined\_S0\_L001\_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プライマー配列先頭のN群を除外するか否か

--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(ILLUMINA | MGI | OTHER | NOCHANGEから選択)

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、インデックス配列だけでなくプライマー配列も使用してデマルチプレックスを行うため、インデックス配列のみを使う場合よりも細かくデマルチプレックスすることが可能です。 したがって、他のプライマーの増幅産物が混入しているデータでも、プライマー配列が十分異なっていればそれらを分けることができます。

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。

出力されたファイルからは、プライマー配列のマッチした部分は除去されています。 これは、その部分はプライマーの塩基配列であって、実際の生物の塩基配列ではないからです。

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。

### 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(OVL | NONから選択)

--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。

### 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、後述するデノイジ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。

### cldenoiseseqdによるデノイジング

以下のコマンドでDADA2 (Callahan et al., 2016) によるデノイジ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(Priorは使用していません)。 Pseudo-pooling法に関してはDADA2の公式Webサイトをご参照下さい。

### 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。

### 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の範囲で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い、または内部標準配列に似た配列を持った生物の配列が含まれている、またはその両方であり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。 合成された内部標準DNAと生物のDNAの区別ができないので、非定量メタバーコーディングとしてもデータを使用することはできません。

### 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

キメラ除去用参照配列データベースは「インストール先/share/claident/uchimedb」にあるため、このフォルダの内容を見ればインストールされている参照配列データベースがわかります。

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)を使用するのは、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。

### 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。

### 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 \  
--solutionvoltable=solutionvoltable.tsv \  
--watervoltable=watervoltable.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リストを記したテキストファイル

--stdconctable

内部標準DNA濃度表のタブ区切りテキスト

--solutionvoltable

抽出DNA溶液量表のタブ区切りテキスト

--watervoltable

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。

## 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

### 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

分子同定用参照配列データベースは「インストール先/share/claident/blastdb」にあるため、このフォルダの内容を見ればインストールされている参照配列データベースがわかります。

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、全部やってしまえばいいでしょう。

### 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。

### QCauto法による分子同定

#### clidentseqによる近隣配列群の取得

以下のコマンドで、QCauto法に基づいて近隣配列をキャッシュデータベースから取得します。

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

コマンドラインオプションの意味は以下の通りです。

--method

使用する分子同定アルゴリズム

--blastdb

使用する分子同定用参照配列データベースまたはキャッシュデータベース

--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、出力ファイルを指定します。

#### classigntaxによる分類群の割り当て

以下のコマンドで、取得した近隣配列の同定情報からLCAアルゴリズム (Huson et al., 2007) を用いて各OTUに分類群を割り当てます。

classigntax \  
--taxdb=animals\_mt\_species\_wsp \  
11\_taxonomy/neighborhoods\_qc\_species\_wsp.txt \  
11\_taxonomy/taxonomy\_qc\_species\_wsp.tsv

コマンドラインオプションの意味は以下の通りです。

--taxdb

使用する参照配列の同定情報データベース(clmakecachedbの--blastdbと一致させる)

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

出力ファイルは、OTUごとに1行の同定結果を記録したタブ区切りテキストになっています。

### 95%-3NN法による分子同定

#### clidentseqによる近隣配列群の取得

以下のコマンドで、95%-3NN法に基づいて近隣配列をキャッシュデータベースから取得します。

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

#### classigntaxによる分類群の割当

以下のコマンドで、取得した近隣配列の同定情報からLCAアルゴリズム (Huson et al., 2007) を用いて各OTUに分類群を割り当てます。

classigntax \  
--taxdb=animals\_mt\_species\_wsp \  
--minnsupporter=1 \  
11\_taxonomy/neighborhoods\_3nn\_species\_wsp.txt \  
11\_taxonomy/taxonomy\_3nn\_species\_wsp.tsv

コマンドラインオプションの意味は以下の通りです。

--minnsupporter

結果を支持する近隣配列数の下限

この方法では、OTUの塩基配列が95%以上一致する参照配列を類似度上位3位タイまで取得して近隣配列とし、LCAアルゴリズム (Huson et al., 2007) を用いて各OTUに分類群を割り当てていますが、95%以上一致する参照配列が1～2本であっても結果を採用するように指定しています(当然、誤同定は生じやすくなります)。

### clmakeidentdbによる分子同定結果の再利用

以下のコマンドを使用してQCauto法による分子同定結果データベースを作成することができます。

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

コマンドラインオプションの意味は以下の通りです。

--append

出力ファイルが既に存在している場合は結果を追加する

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

出力ファイル内には分子同定結果(実際にはclidentseqの結果)が記録されており、clmakecachedbとclidentseqの実行時に--identdbオプションで指定することで、このデータベース内に結果が既にあるOTUにおいて参照配列データベースの検索を飛ばし、無駄な計算を省きます。 なお、手法やデータベースが異なれば分子同定結果は当然ながら異なり得ます。 したがって、clmakeidentdbを--append付きで実行する際は同定手法やデータベースの異なる分子同定結果を混ぜてしまわないように注意が必要です(コマンド側で検証はしていません)。 clmakecachedbで--identdbが指定されると、既に結果が存在するかどうかの確認にしか使用されませんが、clidentseqで--identdbが指定された場合、その分子同定結果データベース内の結果が使用されるため、同定手法やデータベースが一致していなくてはなりません(コマンド側で検証はしていません)。

### clmergeassignによる複数の分子同定結果のマージ

ここまでの解析によって、OTUごとに少なくともQCauto法による分子同定結果と95%-3NN法による分子同定結果が得られているはずです。 複数のデータベースでそれぞれ分子同定を行い、同一のOTUに対してより多くの分子同定結果が得られている場合もあるでしょう。 そのような場合、それらの結果からOTUごとに「最も低レベルの分類階層まで同定できているものを採用する」という形で同定結果をマージすることができます。 下記コマンドを実行すると、より保守的で誤同定が少ないと考えられるQCauto法の結果を優先しつつ、95%-3NN法でQCauto法の結果と矛盾せず、より低レベルの分類階層まで同定できていたら採用する、という形で結果をマージできます(95%-3NN法の結果がより低レベルの分類階層まで同定できていても、QCauto法の結果と矛盾するなら却下します)。

clmergeassign \  
--preferlower \  
--priority=descend \  
11\_taxonomy/taxonomy\_qc\_species\_wsp.tsv \  
11\_taxonomy/taxonomy\_3nn\_species\_wsp.tsv \  
11\_taxonomy/taxonomy\_merged.tsv

コマンドラインオプションの意味は以下の通りです。

--preferlower

より低レベルの分類階層まで同定できている結果を優先的に採用する

--priority

入力ファイルの優先順位(ASCEND | DESCEND | EQUAL | 式による指定から選択)

式は、入力ファイルに0から始まる数値を割り振り、「0<1=2<3<4」という風に指定します。

この優先順位は--preferlowerよりも優先されます

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

--priority=descendを指定している場合、入力ファイル群は後の方よりも最初の方が優先されます。

### clfillassignによる分子同定結果の穴埋め

classigntaxの出力は、そのままでは同定情報のない分類階層は空欄のままとなっています。 そこで、以下のコマンドでそのような空欄を全て埋めることができます。

clfillassign \  
--fullfill=enable \  
11\_taxonomy/taxonomy\_merged.tsv \  
11\_taxonomy/taxonomy\_merged\_filled.tsv

コマンドラインオプションの意味は以下の通りです。

--fullfill

ファイル中に存在しない分類階層も含めてClaidentがサポートしている全分類階層を穴埋めするか否か(ENABLE | DISABLEから選択)

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

穴埋めは、より低レベルの分類階層の結果が存在する場合はその値で、より低レベルの分類階層の結果が存在しない場合は最も低レベルの分類階層の結果に「unidentified」を付加たもので行います。 つまり、orderが「Foo」でinfraorderが「Bar」、その中間のsuborderが空欄の場合、suborderは「Bar」になり、parvorder以下の分類階層が全て空欄ならそれらは「unidentified Bar」となります。

## OTU組成表の作成

ここで言うOTU組成表とは、各サンプルにおける各OTUのリード数の表のことを指します。 以下のような形式で表せるものです。

samplename OTU1 OTU2 OTU3 OTU4  
Sample1 3813 130 1949 34959  
Sample2 18389 19 194 1948  
Sample3 18 1 148 184

この表を元データとして、統計的な解析を行うことになります。 ここでは、実際に統計的な解析に入る前に必要な前処理について説明します。

その前に、以下のコマンドで作業ディレクトリにOTU組成表の出力ディレクトリを作成しておきます。

mkdir 12\_community

また、加工の出発点となるOTU組成表は実は既に「10\_decontaminated/decontaminated.tsv」として存在しているため、以下のコマンドでこれを先程作成したディレクトリにコピーしておきます。

cp \  
10\_decontaminated/decontaminated.tsv \  
12\_community/sample\_otu\_matrix\_all.tsv

### clfiltersumによるOTU組成表の加工

以下のコマンドで、内部標準OTUのみの表を作成することができます(他のOTUは除外される)。

clfiltersum \  
--otuseq=standard.fasta \  
12\_community/sample\_otu\_matrix\_all.tsv \  
12\_community/sample\_otu\_matrix\_standard.tsv

コマンドラインオプションの意味は以下の通りです。

--otuseq

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のデータを取り出す

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

以下のコマンドを実行すると、分子同定結果に基づいて、--includetaxaで指定した分類群(ここでは魚類)のOTUの表を作成することができます。

clfiltersum \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--includetaxa=class,Hyperoartia,class,Myxini,class,Chondrichthyes \  
--includetaxa=superclass,Actinopterygii,order,Coelacanthiformes \  
--includetaxa=subclass,Dipnomorpha \  
12\_community/sample\_otu\_matrix\_all.tsv \  
12\_community/sample\_otu\_matrix\_fishes.tsv

コマンドラインオプションの意味は以下の通りです。

--taxfile

分子同定結果のタブ区切りテキストファイル(classigntaxの出力フォーマットのもの)

--includetaxa

該当する分類群名のOTUのデータを取り出す

分類群名を検索する分類階層を限定することも可能

複数指定可能

下記のように--includetaxaを--excludetaxaに置き換えることで、魚類以外のOTUの表を作成できます。

clfiltersum \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--excludetaxa=class,Hyperoartia,class,Myxini,class,Chondrichthyes \  
--excludetaxa=superclass,Actinopterygii,order,Coelacanthiformes \  
--excludetaxa=subclass,Dipnomorpha \  
12\_community/sample\_otu\_matrix\_all.tsv \  
12\_community/sample\_otu\_matrix\_nonfishes.tsv

同じことを別のやり方でやってみます。 下記のコマンドでは、魚類のOTUの表からOTU名だけを取り出して「12\_community/fishotus.txt」に保存しています。

head -n 1 12\_community/sample\_otu\_matrix\_fishes.tsv \  
| perl -ne '@row=split(/\t/);shift(@row);print(join("\n",@row)."\n");' \  
> 12\_community/fishotus.txt

clfiltersumには、与えたテキストファイルに名前が含まれていないOTUを取り出すオプションがあるので、先程作成したファイルを使用して下記のように魚類以外のOTUの表を作成することができます。

clfiltersum \  
--negativeotulist=12\_community/fishotus.txt \  
12\_community/sample\_otu\_matrix\_all.tsv \  
12\_community/sample\_otu\_matrix\_nonfishes2.tsv

コマンドラインオプションの意味は以下の通りです。

--negativeotulist

除外するOTU名のリストを記したテキストファイル

### clrarefysumによるOTU組成表のカバレッジベースレアファクション

OTU組成表があれば群集生態学解析はできますが、このままではサンプル間のカバレッジ(サンプリング調査の網羅具合)にばらつきがあるため、本来種数の少ない高カバレッジのサンプルの方が本当は種数が多い低カバレッジのサンプルよりも種数が多いと誤判定してしまいかねません。 そこで、サンプル間でカバレッジを揃えることで、このような問題を回避する処理がカバレッジベースレアファクションです (Chao and Jost, 2012) 。 なお、レアファクションが「レアファクションしたOTU組成表を得る」ことを指す場合と「レアファクションカーブを得る」ことを指す場合がありますが、本章では前者を指すものとお考え下さい。

カバレッジベースレアファクションを行う手法としては、「そのサンプルで一度しか観測されていないOTU (シングルトン)の数」と「そのサンプルで二度しか観測されていないOTU (ダブルトン)の数」に基づいてカバレッジを推定して行う方法があります (Chao and Jost, 2012) 。 しかし、メタバーコードデータではシーケンスエラーが大量に存在するために、これらの数が十分信用できるものとは考えられていません (Chiu and Chao, 2016) 。 デノイジングしたデータなら問題ないのではとも思えるかもしれませんが、その証拠も十分でないのが現状です。 Chiu and Chao (2016) はそのようなシーケンスエラーのあるデータでもシングルトン数を修正する方法を提案しており、metagMiscというRパッケージのphyloseq\_coverage\_raref()関数でcorrect\_singletonsを有効にしてレアファクションすることで、この方法が適用できます。

ここで、 はカバレッジそのものと捉えることができます (Chao and Jost, 2012) 。 これに基づいて、Claidentではレアファクションカーブの端点の傾きをサンプル間で揃えるレアファクションをサポートしています。 シングルトン数の影響が薄くなるレアファクションカーブの傾きを使用しているため、シングルトン数とダブルトン数からカバレッジを推定する方法よりもシーケンスエラーに対して頑健であると期待できます。 ただし計算量は多くなるため、並列化と黄金分割探索法を用いた最適化を行っています。

以下のコマンドは、リード数1000未満のサンプルを除去し、残ったサンプルでそれぞれカバレッジを計算し、カバレッジが最も低いサンプルと同じカバレッジに揃うように全サンプルでレアファクションを行います。 ただし、カバレッジが最も低いサンプルのカバレッジが0.99未満だった場合は0.99に揃え、カバレッジが0.99未満のサンプルは除去します。 レアファクションの際には無作為にリードを捨てることになるため、反復すれば結果が変動する可能性があります。 そこで、レアファクションを10反復行い、それぞれ結果を保存します。

clrarefysum \  
--minpcov=0.99 \  
--minnread=1000 \  
--nreps=10 \  
--numthreads=NumberOfCPUcores \  
12\_community/sample\_otu\_matrix\_all.tsv \  
12\_community/sample\_otu\_matrix\_all\_rarefied

コマンドラインオプションの意味は以下の通りです。

--minpcov

揃えるカバレッジの下限(下回るサンプルは捨てる)

--minnread

レアファクション前のリード数下限(下回るサンプルは捨てる)

--nreps

レアファクションの反復数

コマンドラインオプションに引き続いて、入力ファイル、出力ファイルの接頭辞を指定します。

レアファクションが終わったら、以下のコマンドにより10反復全てで内部標準OTUのみを取り出します。

for n in `seq -w 1 10`  
do clfiltersum \  
--otuseq=standard.fasta \  
12\_community/sample\_otu\_matrix\_all\_rarefied$n.tsv \  
12\_community/sample\_otu\_matrix\_standard\_rarefied$n.tsv  
done

以下のコマンドでは魚類OTUのみを取り出します。

for n in `seq -w 1 10`  
do clfiltersum \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--includetaxa=class,Hyperoartia,class,Myxini,class,Chondrichthyes \  
--includetaxa=superclass,Actinopterygii,order,Coelacanthiformes \  
--includetaxa=subclass,Dipnomorpha \  
12\_community/sample\_otu\_matrix\_all\_rarefied$n.tsv \  
12\_community/sample\_otu\_matrix\_fishes\_rarefied$n.tsv  
done

以下のコマンドでは魚類以外のOTUを取り出します。

for n in `seq -w 1 10`  
do clfiltersum \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--excludetaxa=class,Hyperoartia,class,Myxini,class,Chondrichthyes \  
--excludetaxa=superclass,Actinopterygii,order,Coelacanthiformes \  
--excludetaxa=subclass,Dipnomorpha \  
12\_community/sample\_otu\_matrix\_all\_rarefied$n.tsv \  
12\_community/sample\_otu\_matrix\_nonfishes\_rarefied$n.tsv  
done

上記の例では全分類群のOTU組成表を用いてカバレッジベースレアファクションを行い、レアファクション後に魚類OTUと魚類以外のOTUに分けていますが、最初から魚類以外に興味がない場合や、事前知識により魚類以外はコンタミネーションの可能性が高いと思われる場合、魚類のみのOTU組成表を用いてカバレッジベースレアファクションを行う方が良いかもしれません。

metagMiscにしろClaidentにしろ、これらのカバレッジベースレアファクションで行えるのはあくまで「群集に対するシーケンシングカバレッジの均一化」に過ぎないことは注意が必要です。 「採水した水の、群集に対するカバレッジの均一化」や「濾過フィルター上に捕集したDNAの、群集に対するカバレッジの均一化」や「PCRに投入するDNA溶液の、群集に対するカバレッジの均一化」はなされていません。 メタバーコーディングではサンプリング、つまり「一部を取り出す」ステップが多数存在するため、均一性が問題になるのはシーケンシングカバレッジだけではありません。 しかし、それらは全て飽和している(カバレッジ1.0)という仮定のもとでこの先の解析は行われます。 もし何か異常な結果が得られた際には、この仮定が満たされていない可能性について検討すべきかもしれません。

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

以下のコマンドでは、予め濃度がわかっている内部標準DNAリード数に基づいて他のOTUの環境水サンプル中のDNA濃度を推定します。

clestimateconc \  
--stdtable=12\_community/sample\_otu\_matrix\_standard.tsv \  
--stdconctable=stdconctable.tsv \  
--solutionvoltable=solutionvoltable.tsv \  
--watervoltable=watervoltable.tsv \  
--numthreads=NumberOfCPUcores \  
12\_community/sample\_otu\_matrix\_fishes.tsv \  
12\_community/sample\_otu\_matrix\_fishes\_concentration.tsv

コマンドラインオプションの意味は以下の通りです。

--stdtable

内部標準OTUリード数表のタブ区切りテキスト(入力ファイルに内部標準OTUリード数が含まれている場合は不要)

--stdconctable

内部標準DNA濃度表のタブ区切りテキスト

--solutionvoltable

抽出DNA溶液量表のタブ区切りテキスト

--watervoltable

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、入力ファイル、出力ファイルを指定します。

10反復のレアファクションを行ったデータでもそれぞれDNA濃度を推定するには、以下のコマンドを実行します。

for n in `seq -w 1 10`  
do clestimateconc \  
--stdtable=12\_community/sample\_otu\_matrix\_standard\_rarefied$n.tsv \  
--stdconctable=stdconctable.tsv \  
--solutionvoltable=solutionvoltable.tsv \  
--watervoltable=watervoltable.tsv \  
--numthreads=NumberOfCPUcores \  
12\_community/sample\_otu\_matrix\_fishes\_rarefied$n.tsv \  
12\_community/sample\_otu\_matrix\_fishes\_rarefied$n\_concentration.tsv  
done

カバレッジの揃っていないデータでは、推定されるDNA濃度の信頼性がサンプル間でばらつきます。 DNA濃度情報しかないデータからは値の信頼性のばらつきを考慮した解析を行うことはできないので、DNA濃度を利用した分析の際にはレアファクションしてから推定したDNA濃度データを使用する方が良いことが多いのではないでしょうか。 ただ、分析方法によってはレアファクション前の元データから推定したDNA濃度データの方が適している場合もあるかもしれません。

### OTU組成表からの種組成表の作成

OTU組成表は群集生態学解析には適していますが、作図などの際には種組成表や属組成表の方がわかりやすいことがあるでしょう。 そのような場合、OTU組成表と分子同定結果から、種組成表や属組成表を作成することができます。 以下のコマンドでは、魚類のOTU組成表から種組成表を作成しています。

clsumtaxa \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--targetrank=species \  
--taxnamereplace=enable \  
--fuseotu=enable \  
--numbering=enable \  
--sortkey=abundance \  
12\_community/sample\_otu\_matrix\_fishes.tsv \  
12\_community/sample\_species\_matrix\_fishes.tsv

コマンドラインオプションの意味は以下の通りです。

--taxfile

分子同定結果のタブ区切りテキストファイル(classigntaxの出力フォーマットのもの)

--targetrank

指定した分類階層の情報を使用する

--taxnamereplace

出力OTU名内で使用されているスペースやコロンをアンダーバーに置換(ENABLE | DISABLEから選択)

--fuseotu

分類群名が同じOTUをまとめるか否か(ENABLE | DISABLEから選択)

まとめない場合は出力OTU名を「入力OTU名:分類群名」とし、組成の内容は維持する

ただし--taxnamereplaceが有効の場合は出力OTU名は「入力OTU名\_分類群名」となる

--numbering

出力OTU名にソート順で番号を接頭辞として付加するか否か(ENABLE | DISABLEから選択)

出力OTUが100ある場合は001～100という風に幅を揃えた番号をコロン「:」で区切って付加する

--taxnamereplaceが有効の場合はコロンではなくアンダーバー「\_」で区切って付加する

--fuseotuと--taxnamereplaceが有効の場合は「番号\_分類群名」となる

--fuseotuが有効、--taxnamereplaceが無効の場合は「番号:分類群名」となる

--fuseotuが無効、--taxnamereplaceが有効の場合は「番号\_入力OTU名\_分類群名」となる

--fuseotuと--taxnamereplaceが無効の場合は「番号:入力OTU名:分類群名」となる

--sortkey

ソート順を決めるキー(ABUNDANCE | RANKNAMEから選択)

RANKNAMEは「familyname」、「classname」、「“species group name”」などとする

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

以下のコマンドでは、DNA濃度を値とするOTU組成表から種組成表を作成しています。

clsumtaxa \  
--taxfile=11\_taxonomy/taxonomy\_merged\_filled.tsv \  
--targetrank=species \  
--taxnamereplace=enable \  
--taxranknamereplace=enable \  
--fuseotu=enable \  
--numbering=enable \  
--sortkey=abundance \  
12\_community/sample\_otu\_matrix\_fishes\_concentration.tsv \  
12\_community/sample\_species\_matrix\_fishes\_concentration.tsv

なお、--fuseotuを有効化した場合、分類群名だけでOTUがまとめられてしまうため、--targetrank=speciesであっても「unidentified 高次分類群名」という種が存在し、これには多数の種がまとめられてしまう可能性があります。 これは低レベルの分類階層が同定できなかったOTUはclfillassignで「unidentified 高次分類群名」としたためです。 したがって、複数の種が誤ってまとめられたOTUを含む種組成表となってしまいます。 このような種組成表は作図に使用することはできますが、統計的分析にはASVや配列の類似度に基づいてクラスタリングを行ったOTUを単位とするOTU組成表を使用するようにしましょう。

## OTU組成表を用いた群集生態学解析に向けて

ここまでの内容で群集生態学解析に必要なOTU組成表が得られますが、未レアファクションのリード数データ、レアファクション済リード数データ、未レアファクションのDNA濃度データ、レアファクション済DNA濃度データの少なくとも4種類があるはずです。 これらは目的や解析手法に応じて適宜使い分ける必要があります。

まず、レアファクションカーブやヒル数(有効種数) (Chao et al., 2014) の推定・描画には未レアファクションのリード数データを用います。 以下のRパッケージが役に立つでしょう。

* vegan <https://github.com/vegandevs/vegan>
* iNEXT <https://github.com/JohnsonHsieh/iNEXT> (Hsieh et al., 2016)

レアファクション済リード数データはサンプル間での定量性を必要としないほとんどの分析(クラスター分析・NMDS・PerMANOVA・群集系統学解析)に利用できます。 以下のRパッケージについて調べることをお勧めします。

* vegan <https://github.com/vegandevs/vegan>
* picante <https://cran.r-project.org/web/packages/picante/> (Kembel et al., 2010)
* MicEco <https://github.com/Russel88/MicEco>
* bipartite <https://github.com/biometry/bipartite>
* pvclust <https://github.com/shimo-lab/pvclust>
* mpmcorrelogram <https://cran.r-project.org/web/packages/mpmcorrelogram/>

DNA濃度データはサンプル間での定量性が必要な時系列因果推論に使用することができます。 その代わり、整数値を要求する手法を適用することができません。 以下のRパッケージで時系列因果推論を行うことができます。

* rEDM <https://ha0ye.github.io/rEDM/> (Ye and Sugihara, 2016)
* rUIC <https://github.com/yutakaos/rUIC> (Osada et al., 2023)

# 引用文献

Callahan, B.J., McMurdie, P.J., Rosen, M.J., Han, A.W., Johnson, A.J.A., Holmes, S.P., 2016. DADA2: High-resolution sample inference from Illumina amplicon data. Nature Methods 13, 581–583. <https://doi.org/10.1038/nmeth.3869>

Chao, A., Gotelli, N.J., Hsieh, T.C., Sander, E.L., Ma, K.H., Colwell, R.K., Ellison, A.M., 2014. Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. Ecological Monographs 84, 45–67. <https://doi.org/10.1890/13-0133.1>

Chao, A., Jost, L., 2012. Coverage-based rarefaction and extrapolation: Standardizing samples by completeness rather than size. Ecology 93, 2533–2547. <https://doi.org/10.1890/11-1952.1>

Chiu, C.-H., Chao, A., 2016. Estimating and comparing microbial diversity in the presence of sequencing errors. PeerJ 4, e1634. <https://doi.org/10.7717/peerj.1634>

Edgar, R.C., 2016. UCHIME2: Improved chimera prediction for amplicon sequencing. <https://doi.org/10.1101/074252>

Edgar, R.C., 2010. Search and clustering orders of magnitude faster than BLAST. Bioinformatics 26, 2460–2461. <https://doi.org/10.1093/bioinformatics/btq461>

Edgar, R.C., Flyvbjerg, H., 2015. Error filtering, pair assembly and error correction for next-generation sequencing reads. Bioinformatics 31, 3476–3482. <https://doi.org/10.1093/bioinformatics/btv401>

Edgar, R.C., Haas, B.J., Clemente, J.C., Quince, C., Knight, R., 2011. UCHIME improves sensitivity and speed of chimera detection. Bioinformatics 27, 2194–2200. <https://doi.org/10.1093/bioinformatics/btr381>

Esling, P., Lejzerowicz, F., Pawlowski, J., 2015. Accurate multiplexing and filtering for high-throughput amplicon-sequencing. Nucleic Acids Research 43, 2513–2524. <https://doi.org/10.1093/nar/gkv107>

Hsieh, T.C., Ma, K.H., Chao, A., 2016. iNEXT: An R package for rarefaction and extrapolation of species diversity (Hill numbers). Methods in Ecology and Evolution 7, 1451–1456. <https://doi.org/10.1111/2041-210X.12613>

Huson, D.H., Auch, A.F., Qi, J., Schuster, S.C., 2007. MEGAN analysis of metagenomic data. Genome Research 17, 377–386. <https://doi.org/10.1101/gr.5969107>

Kembel, S.W., Cowan, P.D., Helmus, M.R., Cornwell, W.K., Morlon, H., Ackerly, D.D., Blomberg, S.P., Webb, C.O., 2010. Picante: R tools for integrating phylogenies and ecology. Bioinformatics 26, 1463–1464. <https://doi.org/10.1093/bioinformatics/btq166>

Komai, T., Gotoh, R.O., Sado, T., Miya, M., 2019. Development of a new set of PCR primers for eDNA metabarcoding decapod crustaceans. Metabarcoding and Metagenomics 3, e33835. <https://doi.org/10.3897/mbmg.3.33835>

Miya, M., Gotoh, R.O., Sado, T., 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. <https://doi.org/10.1007/s12562-020-01461-x>

Miya, M., Sato, Y., Fukunaga, T., Sado, T., Poulsen, J.Y., Sato, K., Minamoto, T., Yamamoto, S., Yamanaka, H., Araki, H., Kondoh, M., Iwasaki, W., 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. <https://doi.org/10.1098/rsos.150088>

Osada, Y., Ushio, M., Michio, K., 2023. A unified framework for nonparametric causality detection. <https://doi.org/10.1101/2023.04.20.537743>

Rognes, T., Flouri, T., Nichols, B., Quince, C., Mahé, F., 2016. VSEARCH: A versatile open source tool for metagenomics. PeerJ 4, e2584. <https://doi.org/10.7717/peerj.2584>

Sakata, M.K., Kawata, M.U., Kurabayashi, A., Kurita, T., Nakamura, M., Shirako, T., Kakehashi, R., Nishikawa, K., Hossman, M.Y., Nishijima, T., Kabamoto, J., Miya, M., Minamoto, T., 2022. Development and evaluation of PCR primers for environmental DNA (eDNA) metabarcoding of Amphibia. Metabarcoding and Metagenomics 6, e76534. <https://doi.org/10.3897/mbmg.6.76534>

Sato, Y., Miya, M., Fukunaga, T., Sado, T., Iwasaki, W., 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. <https://doi.org/10.1093/molbev/msy074>

Takenaka, M., Yano, K., Suzuki, T., Tojo, K., 2023. Development of novel PCR primer sets for DNA barcoding of aquatic insects, and the discovery of some cryptic species. Limnology 24, 121–136. <https://doi.org/10.1007/s10201-022-00710-5>

Tanabe, A.S., Toju, H., 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. <https://doi.org/10.1371/journal.pone.0076910>

Ushio, M., Fukuda, H., Inoue, T., Makoto, K., Kishida, O., Sato, K., Murata, K., Nikaido, M., Sado, T., Sato, Y., Takeshita, M., Iwasaki, W., Yamanaka, H., Kondoh, M., Miya, M., 2017. Environmental DNA enables detection of terrestrial mammals from forest pond water. Molecular Ecology Resources 17, e63–e75. <https://doi.org/10.1111/1755-0998.12690>

Ushio, M., Furukawa, S., Murakami, H., Masuda, R., Nagano, A.J., 2022. An efficient early-pooling protocol for environmental DNA metabarcoding. Environmental DNA 4, 1212–1228. <https://doi.org/10.1002/edn3.337>

Ushio, M., Murakami, H., Masuda, R., Sado, T., Miya, M., Sakurai, S., Yamanaka, H., Minamoto, T., Kondoh, M., 2018. Quantitative monitoring of multispecies fish environmental DNA using high-throughput sequencing. Metabarcoding and Metagenomics 2, e23297. <https://doi.org/10.3897/mbmg.2.23297>

Ushio, M., Murata, K., Sado, T., Nishiumi, I., Takeshita, M., Iwasaki, W., Miya, M., 2018. Demonstration of the potential of environmental DNA as a tool for the detection of avian species. Scientific Reports 8, 4493. <https://doi.org/10.1038/s41598-018-22817-5>

Ye, H., Sugihara, G., 2016. Information leverage in interconnected ecosystems: Overcoming the curse of dimensionality. Science 353, 922–925. <https://doi.org/10.1126/science.aag0863>

Zhu, T., Sato, Y., Sado, T., Miya, M., Iwasaki, W., 2023. MitoFish, MitoAnnotator, and MiFish Pipeline: Updates in 10 Years. Molecular Biology and Evolution 40, msad035. <https://doi.org/10.1093/molbev/msad035>