

seq2pipe

ローカル LLM による QIIME2 パイプライン自動生成 AI エージェント

Rhizobium-gits Claude (Anthropic)

<https://github.com/Rhizobium-gits/seq2pipe>

2026 年 2 月 23 日

Abstract

本稿では、ローカル大規模言語モデル (LLM) を用いたマイクロバイオーーム解析自動化エージェント **seq2pipe** の設計と実装について報告する。seq2pipe はユーザーが保有する生 FASTQ データを入力として受け取り、データ構造の自動認識・QIIME2 解析パイプラインの設計・実行可能シェルスクリプトの生成・Python による高度な下流解析・可視化・日英 TeX/PDF レポートの自動生成を対話形式で一貫して行う。処理はすべてユーザーのローカル環境で完結し、クラウドサービスや有料 API への依存を排除する。実装には Python 標準ライブラリのみを使用し、Ollama が提供するローカル LLM 推論エンジンを通じて 11 種類のツールによる関数呼び出し (Function Calling) を実現する。起動時の言語選択 (日本語 / 英語)・Python 依存チェック・エラー処理の充実により、研究者が即座に実用できる堅牢なツールを提供する。

1 はじめに

マイクロバイオーーム研究において QIIME2 [1] は 16S rRNA アンプリコンシーケンシングデータの標準的な解析プラットフォームとして広く普及している。しかし、QIIME2 の学習コストは高く、データ形式の理解・適切なパラメータ選択・Docker を介した実行環境の構築など、初心者にとっての障壁が大きい。さらに QIIME2 の出力 (.qzv アーティファクト) の可視化には view.qiime2.org への依存があり、オフライン環境では解析結果の確認も困難である。

近年、大規模言語モデル (LLM) の急速な発展により、自然言語によるコード生成やドメイン特化型の解析支援が実用化されつつある [2]。一方で、従来のクラウド型 LLM サービスはコスト・プライバシー・ネットワーク依存性という問題を持つ。Ollama [4] をはじめとするローカル LLM 実行フレームワークの登場により、これらの課題をクリアしながら高度な言語処理を実現することが可能となった。

本研究では、これらの技術を統合した対話型 AI エージェント **seq2pipe** を開発し、その設計原則・アーキテクチャ・実装詳細を報告する。seq2pipe は QIIME2 解析の自動化に留まらず、Python による下流解析・統計検定・機械学習・可視化、そして日本語・英語両対応の自動レポート生成まで一貫したパイプラインを提供する点で、従来ツールを大きく超える。

2 背景と関連研究

2.1 QIIME2 における解析の複雑性

QIIME2 は、データのインポートからデノイジング (DADA2)・系統樹構築・分類学的解析・多様性解析・差次解析まで、多段階の処理を必要とする。各ステップで適切なコマンドとパラメータを選択するにはシーケンシングのライブラリ構成 (ペアエンド/シングルエンド)・増幅領域 (V1-V3, V3-V4, V4 など)・プライマー配列の知識が不可欠である。

2.2 LLM エージェントの台頭

ReAct フレームワーク [3] に代表されるように、LLM が思考と行動を交互に実行することで複雑なタスクを自律的に処理する「エージェント」パターンが確立されている。関数呼び出し (Function Calling) 機能を持つ LLM は、外部ツールを適切なタイピングで呼び出しながら、ユーザーの意図を達成することができる。

2.3 ローカル LLM の実用化

Ollama は llama.cpp を基盤とし、macOS・Linux・Windows 上で量子化された LLM モデルをシングルバイナリで実行できる。OpenAI の Chat Completions API と互換性のある REST API (/api/chat) を提供しており、関数呼び出しを含む高度な推論が可能である。

3 システム設計

3.1 設計原則

seq2pipe は以下の設計原則に従って開発した。

1. **外部依存ゼロ**: Python 標準ライブラリ (json, urllib, subprocess, pathlib, socket 等) のみを使用し、pip install を不要とする。
2. **ローカル完結**: Ollama のローカル推論エンジンを使用し、インターネット接続を不要とする (初期セットアップを除く)。
3. **クロスプラットフォーム**: macOS・Linux・Windows のすべてで動作する。
4. **安全なコマンド実行**: コマンド実行前にユーザーへの確認を必須とする。
5. **QIIME2 ドメイン知識の内包**: システムプロンプトに QIIME2 の完全なワークフロー知識を埋め込む。
6. **多言語 UI**: 起動時に日本語 / 英語を選択し、AI 応答・自動生成レポートを統一する。
7. **堅牢なエラー処理**: 接続エラー・タイムアウト・空レスポンス・ファイルシステムエラーをすべて適切にキャッチし、ユーザーに分かりやすいメッセージを返す。

3.2 全体アーキテクチャ

図 1 に seq2pipe の全体アーキテクチャを示す。ユーザーは起動スクリプト (launch.sh / launch.bat) を通じて Python エージェント本体 (qiime2_agent.py) を起動する。エージェントは Ollama のローカル API (<http://localhost:11434/api/chat>) と通信し、11 種類のツールを介してファイルシステム・QIIME2 (Docker 経由)・Python 解析エンジンと連携する。



Figure 1: seq2pipe の全体アーキテクチャ

4 実装詳細

4.1 起動シーケンスと多言語 UI

エージェントの起動時に `select_language()` が呼ばれ、ユーザーは日本語 (ja) または英語 (en) を選択する。選択は `LANG` グローバル変数に保存され、`ui()` 関数が全 UI テキストを選択言語で返す。自動生成されるレポートも同じ言語設定を使用する。

また `check_python_deps()` が起動時に `numpy`・`pandas`・`matplotlib`・`seaborn` の存在を `subprocess` 経由で確認し、不足する場合はインストールコマンドを案内する。

4.2 システムプロンプトへのドメイン知識埋め込み

`SYSTEM_PROMPT` 変数に QIIME2 の完全なワークフロー知識を記述した。これには以下の情報が含まれる。

- データ形式の自動判定基準 (*_R1*.fastq.gz パターンによるペアエンド検出など)
- 全 8 ステップの QIIME2 コマンド (インポート～差次解析)
- 増幅領域別推奨パラメータ (V1-V3, V3-V4, V4)
- Docker 実行テンプレート
- メタデータファイル形式仕様
- SILVA 138 分類階層の説明
- 一般的なエラーとトラブルシューティング
- Python 下流解析・自律探索モードの実行指針

この手法により、汎用 LLM モデルが QIIME2 専門家として振る舞うことを可能にした。

4.3 ツール定義と関数呼び出し

Ollama の Function Calling 機能を活用し、11 のツールを JSON Schema 形式で定義した。各ツールは特定の役割を担う (表 1)。

4.4 エージェントループの実装

run_agent_loop() 関数が LLM の応答を受け取り、ツール呼び出しが含まれる場合は順次実行して結果を会話履歴に追加するループを制御する。空レスポンス (content も tool_calls も空) の場合は警告を出して再試行するガード処理を設けている (リスト 1)。

Listing 1: エージェントループの中核部分

```
1 def run_agent_loop(messages: list, model: str):
2     while True:
3         response = call_ollama(messages, model, tools=TOOLS)
4
5         # 空レスポンスのガード
6         if not response["content"] and not response["tool_calls"]]:
7             print("AI からの応答が空でした。再試行します...")
8             continue
9
10        assistant_msg = {"role": "assistant",
11                          "content": response["content"]}
12
13        if response["tool_calls"]:
14            tool_results = []
15            for tc in response["tool_calls"]:
16                fn = tc.get("function", {})
17                result = dispatch_tool(fn["name"],
18                                     fn.get("arguments", {}))
19                tool_results.append({"role": "tool",
```

Table 1: seq2pipe が提供するツール一覧

ツール名	機能
inspect_directory	指定ディレクトリのファイル一覧・サイズを取得し、FASTQ/QZA/メタデータを自動判定する
read_file	テキストファイルの内容を最大 50 行読み込む
write_file	解析スクリプト・マニフェスト・README 等をファイルに書き出す
edit_file	既存ファイルの指定文字列を置換する (old_str new_str)
generate_manifest	FASTQ ディレクトリから QIIME2 マニフェスト TSV を自動生成する
run_command	ユーザー確認後にシェルコマンドを実行する
check_system	Docker・Ollama・Python のインストール状況とディスク容量を確認する
set_plot_config	matplotlib のスタイル・カラー・DPI・フォントサイズ・保存形式を設定する
execute_python	Python コードを実行して解析・可視化を行い、図を FIGURE_DIR に PDF/PNG で保存、ステップを ANALYSIS_LOG に記録する
build_report_tex	ANALYSIS_LOG から TeX を自動構築し、tectonic で日英 PDF レポートをコンパイルする
compile_report	LLM が記述した TeX を受け取りコンパイルする (旧方式)

```

20         "content": result})
21         assistant_msg["tool_calls"] = response["tool_calls"
]
22         messages.append(assistant_msg)
23         messages.extend(tool_results)
24         continue # 次のループで再度 LLM に問い合わせ
25     else:
26         messages.append(assistant_msg)
27         break # テキスト応答のみ -> ループ終了

```

4.5 Ollama API との通信と堅牢なエラー処理

call_ollama() 関数は Ollama の /api/chat エンドポイントに JSON リクエストを送信し、ストリーミングレスポンスを行単位で受信する。Python の urllib.request モジュールのみを使用しており、requests 等の外部パッケージに依存しない。

エラー処理として以下を実装した。

- urllib.error.HTTPError: Ollama サーバーの HTTP エラー (4xx/5xx)

- `urllib.error.URLError + socket.timeout`: タイムアウト (300 秒) とサーバー未起動 (Connection refused) を区別して表示
- `socket.timeout / TimeoutError`: ソケットレベルのタイムアウト

4.6 Python 下流解析と自動レポート生成

`tool_execute_python()` は Python コードを `subprocess.run()` 経由で実行する。実行前に `FIGURE_DIR · PLOT_CONFIG · FIGURE_FORMAT` 変数を自動注入するプレアンブルを付加するため、LLM はファイルパスを意識せずに解析コードを生成できる。実行結果のステップ・図パス・統計結果は `ANALYSIS_LOG` に自動蓄積される。

`tool_build_report_tex()` は `ANALYSIS_LOG` の内容から TeX ソースを Python で完全生成する (LLM を使用しない)。日本語版 (XeLaTeX + xeCJK) と英語版 (pdflatex) を別々に生成し、`tectonic` でコンパイルして PDF を出力する。

4.7 マニフェスト自動生成

`tool_generate_manifest()` は FASTQ ファイルの命名規則 (`_R1_ / _R2_` パターン) を正規表現 (`re.sub(count=1)`) で解析し、ペアエンド・シングルエンド双方の QIIME2 マニフェスト TSV を自動生成する。R2 ファイルの探索は辞書ルックアップ ($O(1)$) で実装し大規模データセットにも対応する。Docker コンテナ内パス (デフォルト: `/data/output`) への変換も自動で行う。ペアが 1 組も見つからない場合はファイルを書かずにエラーを返す。

4.8 クロスプラットフォーム対応

macOS · Linux · Windows の差異を吸収するために以下の対策を実装した。

- **Docker 検出**: macOS では Docker Desktop の固定パス (`/Applications/Docker.app/.../dock`) を優先し、その他の OS では `shutil.which("docker")` を使用する。
- **Windows ANSI カラー**: `os.system("")` を呼ぶことで Windows 10 以降における ANSI エスケープシーケンスを有効化する。
- **起動スクリプトの分離**: macOS/Linux 用 (Bash) と Windows 用 (PowerShell + バッチ) を別々に提供する。

5 解析ワークフロー

seq2pipe が生成する QIIME2 パイプラインは表 2 の 8 ステップで構成される。

SILVA 138 参照データベース [5] を用いた Naive Bayes 分類器により、16S rRNA 増幅領域の分類学的同定を行う。増幅領域 (V1-V3: 27F/338R, V3-V4: 341F/806R, V4: 515F/806R) に応じてプライマートリム長・トランケーション長が自動的に調整される。

QIIME2 解析後、自律探索モードでは以下の 5 フェーズの Python 解析を順番に実行する。

1. **Phase 1 (alpha_diversity)**: Shannon / Simpson / Chao1 指数の計算・可視化・統計検定
2. **Phase 2 (beta_diversity)**: Bray-Curtis PCoA + PERMANOVA

Table 2: 生成される QIIME2 解析パイプラインの概要

STEP	処理	主要コマンド
1	データインポート	qiime tools import
2	クオリティ確認	qiime demux summarize
3	デノイジング (ASV 生成)	qiime dada2 denoise-paired
4	フィーチャーテーブル確認	qiime feature-table summarize
5	系統樹構築	qiime phylogeny align-to-tree-mafft-fasttree
6	分類学的解析	qiime feature-classifier classify-sklearn
7	多様性解析	qiime diversity core-metrics-phylogenetic
8	差次解析 (オプション)	qiime composition ancombc

3. **Phase 3 (taxonomy):** 門・属レベルの分類組成 stacked bar + heatmap
4. **Phase 4 (differential_abundance):** 全 ASV の Mann-Whitney U + BH 補正 + volcano plot
5. **Phase 5 (machine_learning):** Random Forest 5-fold CV + feature importance

全フェーズ完了後、`build_report_tex` が `ANALYSIS_LOG` から日英 TeX/PDF レポートを自動生成する。

6 対応モデルと性能比較

seq2pipe は Ollama で動作する任意の LLM と組み合わせて使用できる。表 3 に推奨モデルとその特性を示す。

Table 3: 対応モデル一覧

モデル	必要 RAM	モデルサイズ	特徴
qwen2.5-coder:7b	8 GB 以上	約 4.7 GB	コード生成特化 (推奨)
qwen2.5-coder:3b	4 GB 以上	約 1.9 GB	軽量・高速
llama3.2:3b	4 GB 以上	約 2.0 GB	汎用・対話能力高め
qwen3:8b	16 GB 以上	約 5.2 GB	最高品質・推論能力高い

7 考察と今後の課題

7.1 達成された目標

seq2pipe は以下の目標を達成した。

- Python 標準ライブラリのみを用いたゼロ依存実装
- ローカル LLM による完全オフライン動作

- 3 OS (macOS / Linux / Windows) への完全対応
- FASTQ データ構造の自動認識と適応的パイプライン生成
- QIIME2 コマンドを Docker 経由で安全に実行するコマンド確認機構
- Python による 5 フェーズの下流解析 (多様性・分類・差次・機械学習)
- ANALYSIS_LOG からの日英 TeX/PDF レポート自動生成
- 日本語 / 英語の起動時言語選択 UI
- 接続エラー・タイムアウト・空レスポンス・ファイルシステムエラーへの堅牢な処理

7.2 現在の制限

- LLM の生成するコマンドの正確性はモデルに依存し、誤ったパラメータが生成される可能性がある。
- 大規模データセット (100 サンプル以上) に対するパフォーマンスは未検証である。
- SILVA 138 参照ファイルのダウンロードには約 30 GB のディスク容量と数時間を要する。
- TeX レポートのコンパイルには tectonic のインストールが必要である。

7.3 今後の課題

- 生成コマンドの自動検証機構の実装
- Web UI の提供 (Gradio や Streamlit との統合)
- ITS / 18S rRNA など他のマーカー遺伝子への対応
- パイプライン実行結果のフィードバックループ実装
- 並列ツール実行による処理の高速化

8 おわりに

本稿では、ローカル LLM と QIIME2 を統合した自動解析エージェント seq2pipe の設計・実装を報告した。Python 標準ライブラリのみによるゼロ依存設計と、クロスプラットフォーム対応・多言語 UI・Python 下流解析・自動レポート生成により、インターネット接続不要の環境でも研究者がマイクロバイオーーム解析を端から端まで自動化できる実用的なツールを実現した。

seq2pipe はオープンソース(MIT ライセンス)として公開されており、<https://github.com/Rhizo> からアクセスできる。

8 References

- [1] Bolyen, E., et al. (2019). *Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2*. Nature Biotechnology, 37, 852–857.
- [2] Brown, T. B., et al. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems, 33, 1877–1901.
- [3] Yao, S., et al. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models*. International Conference on Learning Representations (ICLR 2023).
- [4] Ollama (2023). *Ollama: Get up and running with large language models locally*. <https://ollama.com/>
- [5] Quast, C., et al. (2013). *The SILVA ribosomal RNA gene database project: improved data processing and web-based tools*. Nucleic Acids Research, 41(D1), D590–D596.
- [6] Callahan, B. J., et al. (2016). *DADA2: High-resolution sample inference from Illumina amplicon data*. Nature Methods, 13(7), 581–583.