seq2pipe

A Local LLM-Powered AI Agent for Automated QIIME2 Pipeline Generation

Rhizobium-gits Claude (Anthropic)

`https://github.com/Rhizobium-gits/seq2pipe`

February 23, 2026

Abstract

We present **seq2pipe**, an interactive AI agent that automates end-to-end microbiome analysis by combining a locally running large language model (LLM) with the QIIME2 bioinformatics platform. Given raw FASTQ sequencing data, seq2pipe automatically inspects the data structure, designs an appropriate QIIME2 analysis pipeline, generates ready-to-execute shell scripts, performs Python-based downstream analysis (diversity, taxonomy, differential abundance, machine learning), saves all figures as PDF, and produces bilingual (Japanese/English) TeX/PDF reports — all through natural language dialogue. The entire workflow runs on the user's local machine, eliminating dependencies on cloud services or paid APIs. The implementation uses only Python's standard library and leverages Ollama's local LLM inference engine to orchestrate 11 specialized tools via function calling. A startup language selector (Japanese/English), automatic Python dependency checking, and comprehensive error handling make seq2pipe immediately usable by researchers without bioinformatics backgrounds.

1 Introduction

QIIME2 [1] (Quantitative Insights Into Microbial Ecology 2) has become the de facto standard platform for 16S rRNA amplicon sequencing analysis in microbiome research. However, QIIME2 carries a steep learning curve: users must understand multiple data formats, select appropriate parameters for denoising algorithms (DADA2), manage Docker-containerized execution environments, and interpret multi-step analysis outputs. These barriers make QIIME2 inaccessible to many researchers, particularly those without bioinformatics backgrounds. Furthermore, visualization of QIIME2 outputs (`.qzv` artifacts) typically requires an online viewer (`view.qiime2.org`), creating an additional dependency that is unavailable in offline or restricted-network environments.

The rapid advancement of large language models (LLMs) has opened the door to domain-specific analysis automation through natural language [2]. Yet cloud-hosted LLM services introduce concerns about cost, data privacy, and network availability. The emergence of local LLM inference frameworks such as Ollama [4] allows high-quality language reasoning to be performed entirely on commodity hardware, resolving these concerns.

In this report, we describe the design, architecture, and implementation of **seq2pipe**, an interactive AI agent that integrates local LLM inference with QIIME2 to provide a fully automated, offline-capable microbiome analysis assistant. seq2pipe goes well beyond pipeline generation: it performs Python-based downstream statistical analysis, saves publication-quality figures, and automatically generates bilingual research reports — all within a single conversational interface.

2 Background and Related Work

2.1 Complexity of QIIME2 Analysis

A complete QIIME2 microbiome analysis involves at least eight major steps: data import, quality inspection, denoising (ASV generation via DADA2), feature table summarization, phylogenetic tree construction, taxonomic classification (using SILVA [5]), diversity analysis, and differential abundance analysis. Each step requires careful selection of command-line parameters that depend on the sequencing library configuration (paired-end vs. single-end), the 16S rRNA hypervariable region amplified (V1–V3, V3–V4, V4, etc.), and primer sequences.

2.2 LLM Agents and Function Calling

The ReAct framework [3] established the paradigm of LLMs that interleave reasoning and action — calling external tools at appropriate times to gather information or perform operations that cannot be accomplished through text generation alone. Modern instruction-tuned LLMs supporting function calling can select and invoke pre-defined tools with structured arguments, enabling autonomous task completion.

2.3 Local LLM Inference with Ollama

Ollama wraps the `llama.cpp` inference engine into a single binary distributable across macOS, Linux, and Windows. It exposes a REST API (`/api/chat`) compatible with the OpenAI Chat Completions format, including support for function calling, making it an ideal backend for locally deployed AI agents.

3 System Design

3.1 Design Principles

seq2pipe was developed according to seven core design principles:

1. **Zero external dependencies**: Only Python's standard library (`json`, `urllib`, `subprocess`, `pathlib`, `socket`, etc.) is used, eliminating the need for `pip install`.
2. **Fully local execution**: Ollama's local inference engine is used exclusively, requiring no internet connection after initial setup.
3. **Cross-platform compatibility**: The agent runs on macOS, Linux, and Windows.
4. **Safe command execution**: All shell commands require explicit user confirmation before execution.
5. **Embedded domain knowledge**: The system prompt encodes a complete QIIME2 workflow knowledge base.

6. **Multilingual UI**: The user selects Japanese or English at startup; all AI responses and auto-generated reports follow this preference.
7. **Robust error handling**: Connection errors, timeouts, empty responses, and filesystem errors are all caught and reported with user-friendly messages.

3.2 Overall Architecture

Figure 1 illustrates the overall architecture of seq2pipe. The user launches the Python agent (`qiime2_agent.py`) via platform-specific launch scripts. The agent communicates with the local Ollama API endpoint (`http://localhost:11434/api/chat`) and coordinates 11 specialized tools to interact with the filesystem, QIIME2 (via Docker), and Python analysis libraries.

```
User (language selection: Japanese / English)
  |
  v
[launch.sh / launch.bat]
  |
  v
[qiime2_agent.py] <-----> Ollama (localhost:11434)
  |                              |
  |                          [LLM Model]
  |
  +---> [Tool  1] inspect_directory  (data structure survey)
  +---> [Tool  2] read_file          (read text files)
  +---> [Tool  3] write_file         (write scripts / manifests)
  +---> [Tool  4] edit_file          (partial file editing)
  +---> [Tool  5] generate_manifest  (QIIME2 manifest generation)
  +---> [Tool  6] run_command        (Docker / QIIME2 execution)
  +---> [Tool  7] check_system       (environment check)
  +---> [Tool  8] set_plot_config    (figure style / DPI / format)
  +---> [Tool  9] execute_python     (Python analysis & visualization)
  +---> [Tool 10] build_report_tex   (TeX/PDF report generation)
  +---> [Tool 11] compile_report     (LLM-written TeX compilation)
```

Figure 1: Overall architecture of seq2pipe

4 Implementation Details

4.1 Startup Sequence and Multilingual UI

On launch, `select_language()` prompts the user to choose Japanese (`ja`) or English (`en`). The selection is stored in the `LANG` global variable, and the `ui()` function returns all interface text in the chosen language. Auto-generated reports follow the same language setting.

`check_python_deps()` verifies that numpy, pandas, matplotlib, and seaborn are available at startup via `subprocess`, guiding the user to install any missing packages before analysis begins.

4.2 Embedding Domain Knowledge in the System Prompt

The `SYSTEM_PROMPT` variable contains a comprehensive QIIME2 workflow knowledge base. This includes:

- Automatic data format detection criteria (e.g., paired-end detection via `*_R1*.fastq.gz` patterns)
- Complete QIIME2 commands for all eight analysis steps (import through differential abundance analysis)
- Region-specific recommended parameters for V1–V3 (27F/338R), V3–V4 (341F/806R), and V4 (515F/806R) amplicons
- Docker execution command templates
- Metadata file format specifications
- SILVA 138 taxonomic hierarchy explanation
- Common errors and troubleshooting guidance
- Python downstream analysis and autonomous exploration mode guidelines

By embedding this knowledge directly into the system prompt, a general-purpose code LLM is transformed into a QIIME2 domain expert.

4.3 Tool Definitions and Function Calling

Eleven tools are defined in JSON Schema format compatible with Ollama's function calling interface. Table 1 summarizes the tools and their roles.

4.4 The Agent Loop

The `run_agent_loop()` function controls the core reasoning-action cycle. It sends conversation history to the LLM, detects tool calls in the response, executes them sequentially, appends results to the conversation history, and repeats until the LLM produces a final text-only response. An empty-response guard retries automatically if the LLM returns neither content nor tool calls (Listing 1).

Listing 1: Core agent loop implementation

```python
def run_agent_loop(messages: list, model: str):
    while True:
        response = call_ollama(messages, model, tools=TOOLS)

        # Guard: retry on empty response
        if not response["content"] and not response["tool_calls
    "]:
            print("Empty response from AI. Retrying...")
```

Table 1: Tools provided by seq2pipe

Tool Name	**Function**
`inspect_directory`	Lists files with sizes and automatically identifies FASTQ, QZA, and metadata files
`read_file`	Reads text files (TSV, CSV, Markdown, etc.) up to 50 lines
`write_file`	Writes generated scripts, manifests, and README files to disk
`edit_file`	Replaces a unique string in an existing file (old_str → new_str)
`generate_manifest`	Auto-generates a QIIME2 manifest TSV from a FASTQ directory
`run_command`	Executes shell commands after explicit user confirmation
`check_system`	Verifies Docker, Ollama, Python, and available disk space
`set_plot_config`	Configures matplotlib style, palette, DPI, font size, and output format
`execute_python`	Executes Python code for analysis/visualization, saves figures to `FIGURE_DIR`, and appends steps to `ANALYSIS_LOG`
`build_report_tex`	Programmatically builds TeX from `ANALYSIS_LOG` and compiles bilingual Japanese/English PDF reports via tectonic
`compile_report`	Receives LLM-written TeX source and compiles it to PDF (legacy mode)

```
            continue

        assistant_msg = {"role": "assistant",
                         "content": response["content"]}

        if response["tool_calls"]:
            tool_results = []
            for tc in response["tool_calls"]:
                fn = tc.get("function", {})
                result = dispatch_tool(fn["name"],
                                       fn.get("arguments", {}))
                tool_results.append({"role": "tool",
                                     "content": result})
            assistant_msg["tool_calls"] = response["tool_calls"
    ]
            messages.append(assistant_msg)
            messages.extend(tool_results)
            continue    # Query LLM again with tool results
        else:
            messages.append(assistant_msg)
            break        # Text-only response: end loop
```

4.5 Communication with the Ollama API and Error Handling

The `call_ollama()` function sends JSON requests to Ollama's `/api/chat` endpoint with streaming enabled, and processes server-sent events line by line using only `urllib.request` from Python's standard library.

Comprehensive error handling covers:

- `urllib.error.HTTPError`: Ollama server HTTP errors (4xx/5xx)
- `urllib.error.URLError + socket.timeout`: distinguishes timeouts (300 s) from connection-refused errors (Ollama not running)
- `socket.timeout / TimeoutError`: socket-level timeout fallback

4.6 Python Downstream Analysis and Automatic Report Generation

`tool_execute_python()` executes Python code via `subprocess.run()` with a timeout of 300 seconds. Before execution, a preamble is prepended that injects `FIGURE_DIR`, `PLOT_CONFIG`, and `FIGURE_FORMAT` variables, allowing the LLM to generate analysis code without knowing the exact file paths. Each execution appends a structured entry (step name, figure paths, statistical results) to the global `ANALYSIS_LOG`.

`tool_build_report_tex()` reads `ANALYSIS_LOG` and programmatically generates complete TeX source — no LLM involvement required, making it fast and deterministic. Separate Japanese (XeLaTeX + xeCJK) and English (pdflatex) documents are produced and compiled to PDF via tectonic.

4.7 Automatic Manifest Generation

The `tool_generate_manifest()` function uses regular expressions (`re.sub(count=1)`) to parse FASTQ filenames, automatically generating QIIME2 manifest TSV files for both paired-end and single-end data. R2 file lookup is implemented with a dictionary for O(1) performance, scaling to large datasets. If no R1/R2 pairs are found, the function returns an error without writing an empty manifest. Partial matching rates below 80% trigger an enhanced warning showing the mismatch percentage. Path translation to Docker container-internal paths (default: `/data/output`) is performed automatically.

4.8 Cross-Platform Compatibility

Three platform-specific differences are handled explicitly:

- **Docker detection**: On macOS, the Docker Desktop binary path (`/Applications/Docker.app/.` is checked first; on other platforms, `shutil.which("docker")` is used.
- **Windows ANSI color support**: Calling `os.system("")` activates ANSI escape sequence processing in Windows 10+ terminals.
- **Separated launch scripts**: Bash shell scripts for macOS/Linux and PowerShell + batch files for Windows are provided separately.

5 Analysis Workflow

The QIIME2 pipeline generated by seq2pipe consists of eight steps as summarized in Table 2.

Table 2: Overview of the generated QIIME2 analysis pipeline

Step	Process	Key Command
1	Data import	`qiime tools import`
2	Quality visualization	`qiime demux summarize`
3	Denoising (ASV generation)	`qiime dada2 denoise-paired`
4	Feature table summarization	`qiime feature-table summarize`
5	Phylogenetic tree construction	`qiime phylogeny align-to-tree-mafft-fasttree`
6	Taxonomic classification	`qiime feature-classifier classify-sklearn`
7	Diversity analysis	`qiime diversity core-metrics-phylogenetic`
8	Differential abundance (opt.)	`qiime composition ancombc`

Taxonomic classification is performed using a Naive Bayes classifier trained on the SILVA 138 reference database [5]. Primer trim lengths and truncation positions are automatically adjusted based on the detected hypervariable region (V1–V3: 27F/338R, V3–V4: 341F/806R, V4: 515F/806R).

Following QIIME2 analysis, the autonomous exploration mode executes five phases of Python downstream analysis:

1. **Phase 1 (alpha_diversity)**: Shannon / Simpson / Chao1 indices with statistical tests
2. **Phase 2 (beta_diversity)**: Bray-Curtis PCoA with PERMANOVA
3. **Phase 3 (taxonomy)**: Phylum/genus-level stacked bar charts and heatmaps
4. **Phase 4 (differential_abundance)**: All-ASV Mann-Whitney U with Benjamini-Hochberg correction and volcano plot
5. **Phase 5 (machine_learning)**: Random Forest 5-fold CV with feature importance visualization

Upon completion, `build_report_tex` automatically generates bilingual TeX/PDF reports from `ANALYSIS_LOG`.

6 Supported Models and Performance

seq2pipe is model-agnostic and can be used with any LLM available in Ollama. Table 3 lists recommended models.

The code-generation-optimized `qwen2.5-coder:7b` is recommended as the default model. For RAM-constrained environments (e.g., 8 GB MacBook Air), `qwen2.5-coder:3b` or `llama3.2:3b` provide a lighter-weight alternative with minimal capability trade-off for structured QIIME2 command generation.

Table 3: Supported models

Model	RAM Required	Size	Characteristics
`qwen2.5-coder:7b`	8 GB+	~4.7 GB	Code generation optimized (recommended)
`qwen2.5-coder:3b`	4 GB+	~1.9 GB	Lightweight and fast
`llama3.2:3b`	4 GB+	~2.0 GB	General purpose, strong dialogue
`qwen3:8b`	16 GB+	~5.2 GB	Highest quality, strong reasoning

7 Setup and Launch

7.1 Software Requirements

Table 4 summarizes the required software stack.

Table 4: Software requirements

Software	Purpose	Installation
Python 3.9+	Agent runtime	Typically pre-installed
Ollama	Local LLM inference	Automated via `setup.sh`
Docker Desktop/Engine	QIIME2 execution environment	Manual install
numpy, pandas, etc.	Python downstream analysis	Automated via `setup.sh`
tectonic (optional)	PDF report compilation	`brew install tectonic`

7.2 Launch Sequence

1. Run `setup.sh` (macOS/Linux) or `setup.bat` (Windows) to install Ollama, download the LLM model, and install Python packages.
2. Run `launch.sh` / `launch.bat`: the script verifies Ollama and Docker availability, then starts the agent.
3. Select Japanese or English at the language prompt.
4. Through natural language dialogue, provide the data directory path and specify the desired analyses (taxonomic composition, diversity, differential abundance, etc.).
5. The agent automatically inspects the data, generates a customized script bundle, runs Python analyses, and produces a PDF report.

8 Discussion

8.1 Achieved Goals

seq2pipe achieves the following:

- Zero-dependency implementation using only Python's standard library
- Fully offline operation via local LLM inference

- Complete cross-platform support (macOS / Linux / Windows)
- Automatic recognition of FASTQ data structure and adaptive pipeline generation
- Safe QIIME2 command execution with explicit user confirmation
- Python-based 5-phase downstream analysis (diversity, taxonomy, differential abundance, machine learning)
- Automatic bilingual (Japanese/English) TeX/PDF report generation from `ANALYSIS_LOG` without LLM involvement
- Startup language selection UI (Japanese / English)
- Robust handling of connection errors, timeouts, empty responses, and filesystem errors

8.2 Current Limitations

- The accuracy of generated QIIME2 commands depends on the underlying LLM model quality; incorrect parameters may be produced.
- Performance with large datasets (100+ samples) has not been formally evaluated.
- Initial SILVA 138 classifier training requires approximately 30 GB of disk space and several hours of computation.
- PDF report compilation requires tectonic to be installed separately.

8.3 Future Directions

- Implementation of automated command validation and error correction
- Web UI integration (Gradio or Streamlit)
- Support for additional marker genes (ITS, 18S rRNA)
- Feedback loop incorporating pipeline execution results to iteratively refine parameters
- Parallel tool execution for faster autonomous exploration

9 Conclusion

We have described seq2pipe, an interactive AI agent that integrates local LLM inference with the QIIME2 microbiome analysis platform. By embedding comprehensive QIIME2 domain knowledge into the system prompt and providing 11 specialized tools through Ollama's function calling interface, seq2pipe enables researchers to automate an entire 16S rRNA analysis workflow — from raw FASTQ data through statistical analysis, publication-quality figures, and bilingual research reports — entirely offline and without any cloud dependencies.

seq2pipe is released as open source under the MIT License and is available at: `https://github.com/`

9 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.