# Vibe コーディングから宣言的プログラムへ：体験から学ぶ医療教育アプリ開発手法の提案

中原龍一[1] 高橋康[2] 竹内孔一[3] 長谷井嬢[4]
[1]岡山大学大学院医歯薬学総合研究科　運動器地域健康推進講座 [2]日本電気株式会社
[3]岡山大学大学院環境生命自然科学研究科 [4]岡山大学学術研究院医歯薬学域 医療情報化診療支援技術開発講座
[1] r-nakahara@okayama-u.ac.jp, [2] yas.takahashi@nec.com, [3] takeuc-k@okayama-u.ac.jp, [4] py3g9rcw@s.okayama-u.ac.jp,

## 概要

近年、生成 AI を用いた「Vibe コーディング」により、専門的なプログラミング知識がなくともアプリケーション開発が可能となった。一方で、生成 AI の確率的挙動に起因する潜在的なバグの蓄積や、「プログラムの基礎を学ばなくてもよい」という誤解が広がるという問題も生じている。

本研究では、これら二つの課題に対する解決策として、アプリ開発過程を「宣言的プログラム」として再構成し、それを医療教育アプリ開発に適用した。医療教育アプリの実装は以下で公開している。
https://github.com/duckdbdemo/Demo_DuckDB_Semanticlayer_Web_v2

## 1．はじめに

### 1.1　課題：「Vibe コーディング」の光と影

近年、生成 AI による「Vibe コーディング」が急速に普及している。プログラミングの基礎や動作原理を理解しなくても、現場の感覚と試行錯誤だけでアプリ開発が可能であるため、アプリ開発コストを劇的に下げた。

しかし生成 AI は確率的な挙動を示すため、プログラム内部に小さなバグが蓄積されてしまう。修正コストが積みあがってしまう「ワークスロップ」を生み出してしまう。

更に Vibe コーディングは「プログラムの基礎を学ばなくてもよい」という学習態度を一部で生み出してしまっており、プログラムの基礎知識があれば避けることが可能なバグを生み出す温床となっている。

つまり Vibe コーディングはアプリ開発コストを劇的に低下させたが、**「ワークスロップ」**と**「プログラムの基礎を学ばなくてもよいという勘違い」**という問題を生み出したといえる。

### 1.2　解決策：「宣言的プログラミング」

本研究では、この二つの問題を解決するため「宣言的プログラム」の利用を提案する。これは、宣言的なプログラム生成をデータ解析のように、「プログラムの変更プロセスの連鎖」として捉えなおし、プログラム変更のパイプラインとして記述する手法である。

本手法の発想は、R 言語の tidyverse のような「データ変換の連鎖」をアプリ開発プロセスにも適応できないかという視点から生まれた。具体的には、「アプリ開発後に、同じアプリが生成されるプロンプトをあえて再構築し、編集可能なプログラム変更の連鎖として組みなおす」という工程によって宣言的プログラムとして作り直すことを目指した（図１）。

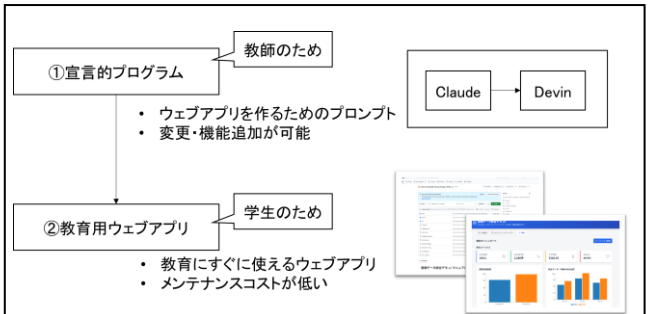


図１：本研究の概要

### 1.3　実践：医療統計教育における

この手法の有効性を検証するために医療統計教育を選んだ。医学統計は伝統的に医療統計と呼ばれる医療内部で教育が完結される教育手法として発展してきた。しかし近年の統計技術の発展に伴い、医療統計教育が対象にしてこなかった、医療以外の領域

で生まれた最新技術を教える必要性が出てきた。特にEコマースを中心として発展してきたクラウド上での統計手法(データ基盤上での拡張されたSQLによる統計パイプ処理や、「自然言語によるデータ探索」や「複数の病院データベースの統合」を可能にするセマンティックレイヤーなど）を教育する必要が生まれている。つまり医療統計が対象としてこなかった領域の統計手法を教える必要が生まれている。

しかし医療データを扱う都合上、ローカルで完結した教育環境を開発する必要がある。クラウドが前提である既存の教材が利用できない。しかし、ローカルでデータ基盤が構築可能な環境（duckDBなど）が生まれたことで、ローカル環境でデータ基盤関連技術を教えることが可能となった。

今回は「ローカルデータ基盤を通じて病院Aと病院Bのデータベースを統合して解析するには、どのような機能が必要なのかを体感できるwebアプリ」を構築したのち、宣言的プログラムに変換したのでその結果を報告する。

## ２．方法

### 2.1 医療統計教育タスクの抽出

**タスク１：二つの病院DBの接続**

二つの異なる構造を持った病院データを結合するタスクは臨床研究でよく遭遇する課題である。しかし、医療教育の現場ではその解決法を教えることは少なく、綺麗に構造化されたデータに対して統計処理を行う教育が主体だった。

そこでダミーデータを用いて異なる構造を持った病院データを、セマンティックレイヤーを通じて結合するというタスクを考えた。

**タスク２：列型DB**

医療データは大規模化しており、従来の行型のデータ処理では困難なほどのデータ量を扱う必要が出てきた。クラウドでは列型DB利用が一般化してきているが、ローカルでの列型DB利用の医療教育はまだ一般的ではない。そこで二つの異なるDBを列型に変換したのちに、セマンティックレイヤーを通じて統計解析を行うというタスクを考えた。

**タスク３：解析コードの可視化**

医療統計ソフトはマウス操作だけで統計操作が可能であるため、統計操作はマウス操作で行うものという認識が強い。しかし現代的な統計ではR言語やSQLのようにコード化されている。コードさえ書けば再現性をもって解析・可視化・統計が可能であることを体験できるタスク。

### 2.2 アプリ化

これらの3種類のタスクを体験するためのコードをPython コードとして作成。Jupyter Notebookで動作を確認。作成したコードをdevinを用いてwebアプリ化し、webアプリでタスク①②③が体験できることを確認した。

### 2.3 宣言プログラム化

必要な機能やライブラリを抽出し、ClaudeでPython コードをJupyter Notebookとして作成し、次にPythonコードを元にDevinでweb化可能なプロンプトリストを作成した。

そのプロンプトをもとに、同じようなアプリが開発できるかどうかを確認した。またプロンプトを分割して機能変更が可能な構成にした。

## ３．結果

### 3.1 アプリ機能

開発した医療教育用webアプリは3つのタブから形成されている。

一つ目のタブは二つの病院DBを可視化、編集する機能。デフォルトでデータがあるため、学生は入力操作をする必要がない。また編集機能があるため、構造が変わってもセマンティックレイヤーを操作すれば結合可能であることなどを体験できる。

二つ目のタブはセマンティックレイヤーの設定と編集の機能。病院Aと病院Bの項目がどのような用語で統合されるかを記述しておけば、可視化・統計の段階で二つの病院の内部表記を考える必要がないことを体験できる。セマンティックレイヤーはデフォルトで記述されているため学生は入力する必要はないが、DBの要素を変更した時には変更が必要なことを体験できる。

三つめのタブはデータの可視化とSQL記述の機能。自動的に統計・可視化が行われるが、その記述

は SQL に起因することが体験できる。工学的には当たり前であるが、医療関係者にとっては UI 操作なしに自動的に統計できるという体験は非常に有効。SQL はデフォルトで記述されており、セマンティックレイヤーで定義した名前で統計が可能であることが体験できる。これも工学的には当たり前であるが医療関係者にとっては非常に教育的。編集機能があるため、自分で統計を記述することも可能。

このように本アプリは「体験を通じて学ぶ」構造を重視した（図２）。

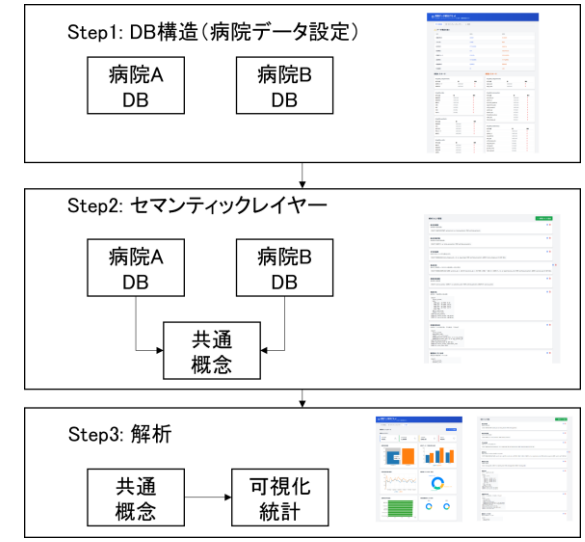


図２：教育用アプリの概要

### 3.2 宣言的プログラム

宣言的プログラムによって同等の機能を持つアプリの再現と、機能追加に成功した（図３）。

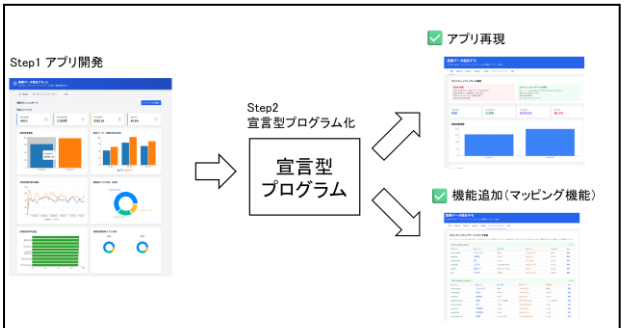


図３：再現性実験

アプリ生成のための宣言的プログラムは大きく分けて二つの構成からなる（図４）。Claude を用いて Python コードを生成する部分と、Devin を用いて web アプリを作る部分である。すべてを Devin で作らずに、あえて二つに分割した理由は、2025 年の時点で Python コードの生成能力は Claude がもっとも高いからである。また web アプリで挙動を体験した後に、Jupyter Notebook の Python コードを見て学んでもらうためである。

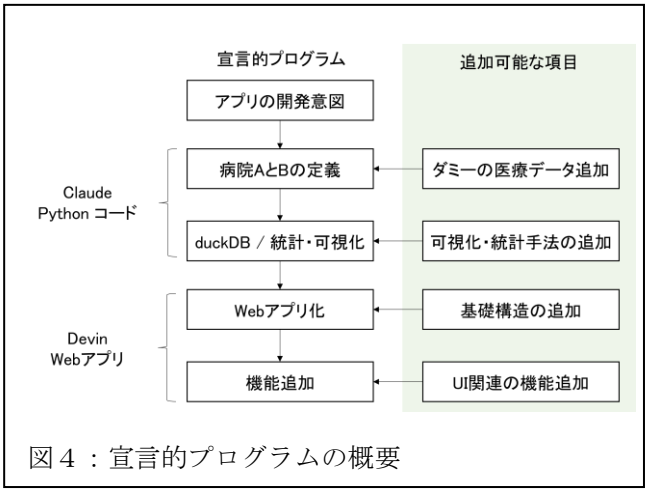


図４：宣言的プログラムの概要

宣言的プログラムはさらに再分割されており、ダミーデータの登録や統計機能の追加、web の UI 機能が拡張可能な構造にした。図のように分割したのは、データ登録や統計機能は前半の Python コードの時点で決定したほうが安定したからである。この分割構造は将来のプログラム支援生成 AI の発展によって変化する可能性がある。

## 4．考察

### 4.1 宣言的プログラムの経済的価値

プログラムよりも、データに価値があり、生データの蓄積だけでなく、生データを使える形に自動的に変換するシステムへの投資に価値があることが、経済学領域における無形資産の研究[1]から判明している。本研究ではプログラムも同じように、生データに対する変換システムと捉えることができないかと考えた。つまり教育用プログラムを作るだけでなく、教育意図に基づいて教育用プログラムを生成する（変換する）システムの開発を行うことで、教育投資効率が改善されるのではないかと考え、その基礎的研究として本研究を位置づけた。

宣言的プログラムの利点は、言葉で書かれているため非エンジニアでも意味が取れる点にある。宣言的プログラムのプロンプトを生成 AI だけでなく、

人間に対しても読みやすくコメントすることで、教育効果を高める可能性がある。

### 4.2 宣言プログラムの技術的背景

宣言的プログラムのアイディアは、R 言語で発展した tidyverse 思想から着想を得た。R 言語での tidyverse の開発過程[2]を振り返ってみると、様々なパッケージを統合し、動詞を共有化したことが重要なステップであったことが判明している。tidyverse の出現以前は、似ているが異なる名前を持つパッケージ群が乱立していることが問題だった。tidyverse はデータ変換でよく利用される変換手法を少数の動詞として集約することでパッケージ間の統合に成功した。データ変換によく利用される機能を、少数の動詞とパラメータとして表現することで、人間の認知負荷が一気に低下した。この手法は R 言語以外にも「データ変換の抽象化概念」として広がった。例として Python のデータ変換処理や、データ基盤上の拡張された SQL、今回のテーマである duckDB や、セマンティックレイヤーを民主化した dbt などがあげられる。

宣言的プログラムの利点はプログラムの基礎概念をもとに記述できる点にある。プログラムの基礎概念を知らずに Vibe コーディングで作成したとしても、宣言的プログラムとして再利用可能な形に変換するためには、プログラムの基礎知識の習得が必要になる。この工程をとることで、最初は Vibe コーディングで気持ちよく何も知らずに作った後に、なぜこれらの機能が成立したのかを宣言的プログラムとして考え直すことができる。また tidyverse のデータ変換のように細かいステップのパイプ処理として記述することで、確率挙動を減らし、問題が発生したとしても探知しやすくなる。

### 4.3 体験してから学ぶ教育

Vibe コーディングは「プログラムの基礎を学ばなくて良い」という勘違いを作ってしまう危険性はある。しかし簡単に作れるという体験によって、プログラムに触れる人が増えているのも事実である。それは否定するべきではないと思う。問題は簡単に作った後に、学びなおさない事にあると思われる。

本研究では、医学統計を学ぶときも、Vibe コーディングのように楽しく学べるアプリが作れないか、ということを考えた。まずサンドボックス的なデモ環境で、様々な要素を自分の意思で変更することで統計結果や可視化の挙動を体験した後の方が、教科書的な教材を学びやすくなる（図５）。体験してから学ぶ仕組みを教育者が容易に構築できるような環境を作ることが本研究の根本目的である。

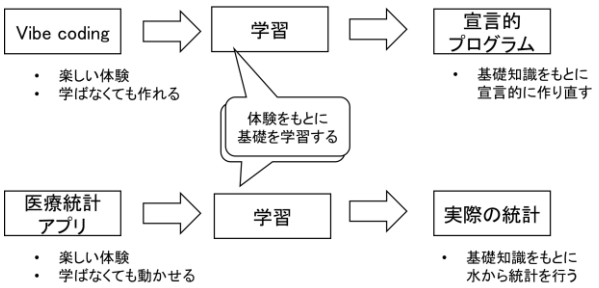


図５：体験してから学ぶ

AI の出現により、時間と根性さえあれば大抵の知識は学べてしまえるようになった。「情報の飽食の時代」とでも呼べるような状態になったといえる。そのような時代にあっては、教育機関は知識の提供だけでなく、知識が体験できる場を作る必要がある。従来はそのようなアプリ開発はコスト面で問題があったが、宣言的プログラムによってコスト問題が解決される可能性がある。

## ５．リミテーション

本研究のリミテーションとして、以下の点が挙げられる。第一に、本手法を実際の学生に利用させた定量的評価を実施していない点である。第二に、著者ら以外の教育関係者による宣言的プログラムを用いた医療アプリ修正の検証が行われていない点である。第三に、Devin や Claude などの特定の生成 AI に依存している点である。

1. Liu, M., et al., *Optimizing Accounting for Data Assets Helpful To Developing Sustainable Regional Economies.* Information Resources Management Journal (IRMJ), 2024. **37**(1): p. 1-17.

2. Wickham, H., *A personal history of the tidyverse*. 2025.