From SAS to Cursor: Vibe-coding into SAS, R, Python in Biometrics

Kevin Lee

Clinvia

Proprietary and Confidential Information – Do Not Copy or Distribute

Disclaimer

The views and opinions presented here represent those of the speaker and should not be considered to represent any companies or organizations.

Agenda

- Introduction of Vibe-coding
- Current Coding in Biometrics: SAS, R, and Python
- The integration of Vibe coding in Biometrics
- Vibe coding tools - Cursor and Windsurf, github copilot
- Customized AI coding agents for Biometrics
- Benefits and Risks
- Future of Stat Programmers in the era of Vibe-coding

Confidential Information – Do Not Copy or Distribute

Where “Vibe-Coding” come from?

“fully giv[ing] in to the vibes, embrac[ing] exponentials, and forget[ting] that the code even exists.”

by Andrej Karpathy,
co-founder of OpenAI
on Feb, 2025

“There's a new kind of coding I call "vibe coding," where you fully give in to the vibes.”
-Andrej Karapathy

Confidential Information – Do Not Copy or Distribute

What is Vibe-Coding?

- Coding with AI
- Vibe : A flow between human logic and AI coding agents
- Programmers do
  - Prompt for the codes
  - Review the codes

The impact is Vibe-Coding?

- Satya Nadella, CEO of Microsoft

  “I’d say maybe 20%, 30% of the code that is inside of our repos today and some of our projects are probably all written by software,”

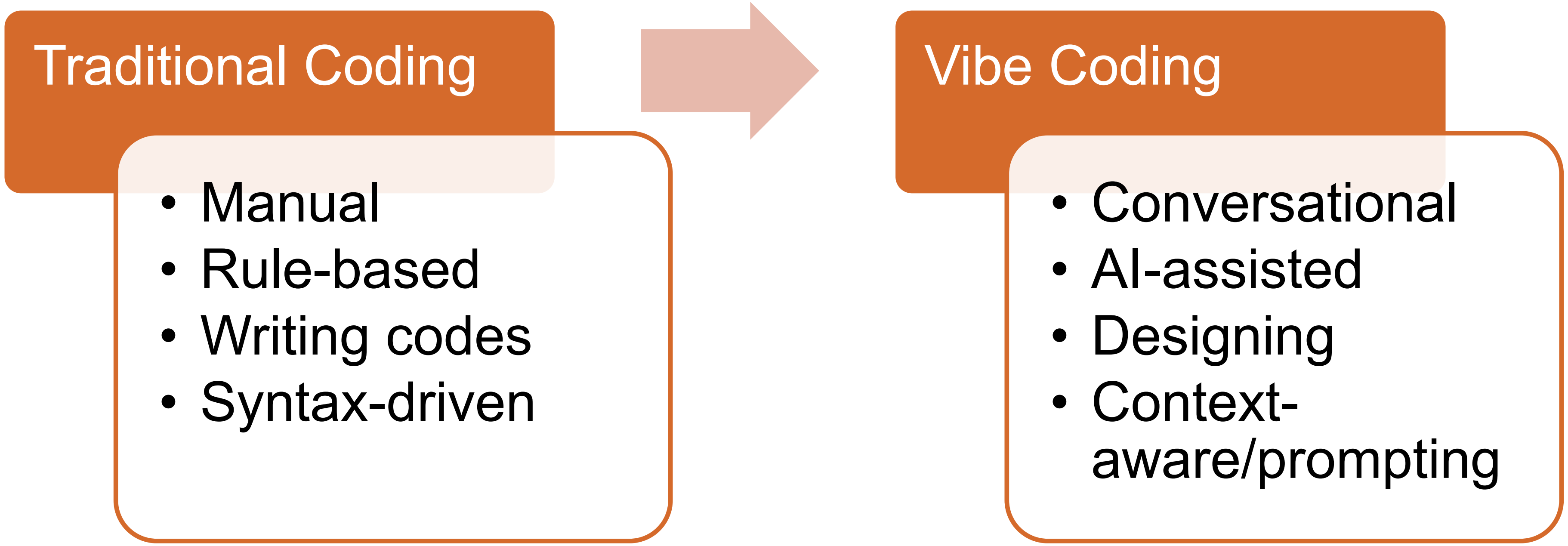
- Sundar Pichai, CEO of Google

  "AI generates more than 30% of Google’s code,“

- Mark Zuckerberg, CEO of Meta

  “Our bet is sort of that in the next year probably…maybe 50% of the development is going to be done by AI, as opposed to people, and then that will just kind of increase from there,”

The Change of Coding



Traditional Coding

- Manual
- Rule-based
- Writing codes
- Syntax-driven

Vibe Coding

- Conversational
- AI-assisted
- Designing
- Context-aware/prompting

Confidential Information – Do Not Copy or Distribute

Triad of Stat Programming in Biometrics

SAS

- Regulatory Backbone
- Majority on SDTM, ADaM, TFL

R

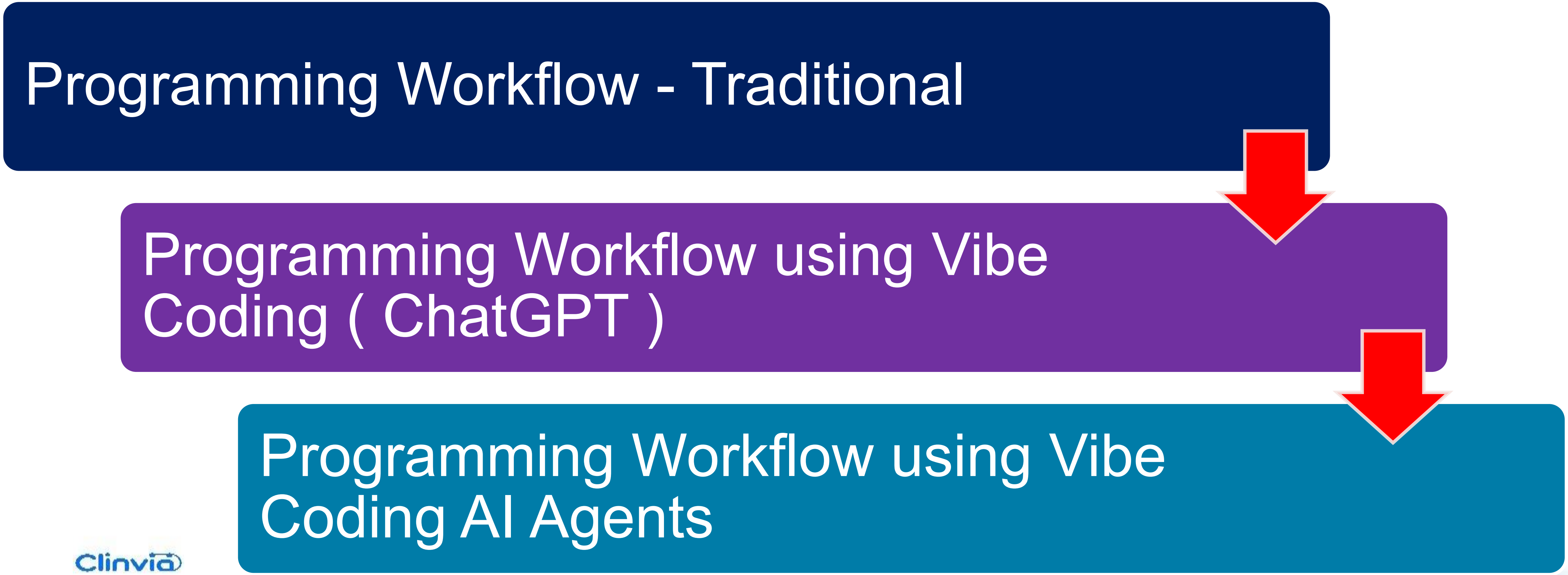
- Data Visualization
- Shiny
- Beginning in SDTM, ADaM, TFL

Python

- Data Pipeline
- AI/ML
- Utilities function

Confidential Information – Do Not Copy or Distribute

Programming Workflow



Programming Workflow - Traditional

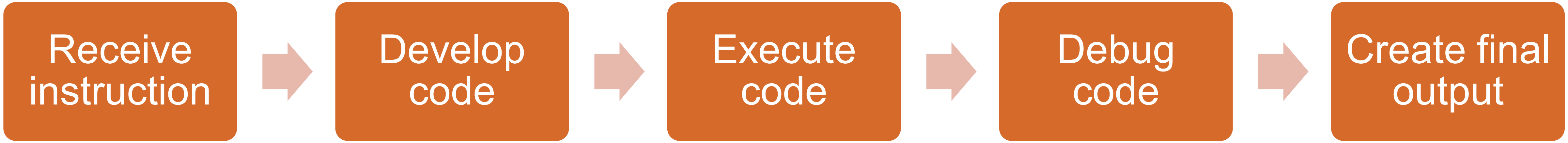
Programming Workflow using Vibe Coding (ChatGPT)

Programming Workflow using Vibe Coding AI Agents

Clinvia

Confidential Information – Do Not Copy or Distribute

Traditional Programming workflow



Receive instruction

Develop code

Execute code

Debug code

Create final output

Confidential Information – Do Not Copy or Distribute

ChatGPT impact on programming

- Code Development
- Code Debugging
- Lowering the boundary of languages (e.g., SAS, R and Python) – vibe coding

ChatGPT: Codes Development – R

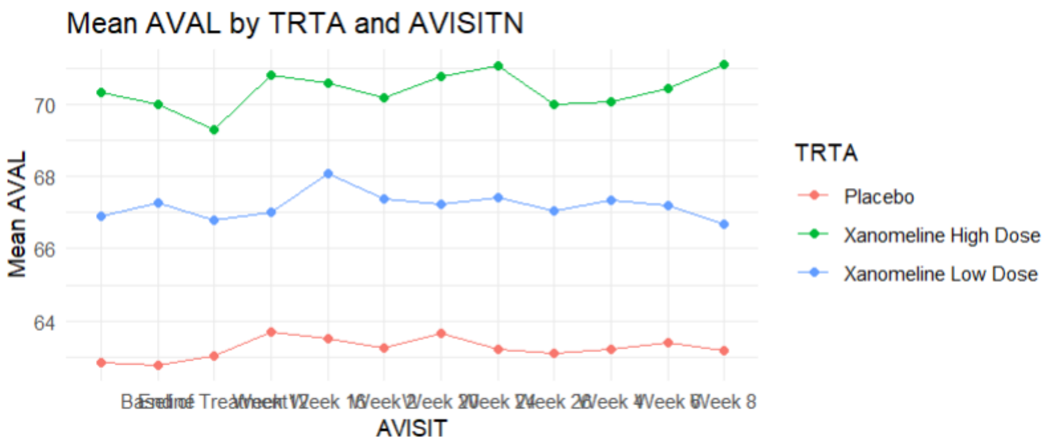
Prompt :

Write a **R script** that groups data by “TRTA" and “AVISIT" columns in "advs.xpt"

and calculates the mean of “AVAL" column for each group at PARAMCD =‘WEIGHT’

Output :

```
library(haven)  # For reading XPT files
library(dplyr)  # For data manipulation

# Read the ADaM dataset
advs <- read_xpt("advs.xpt")

# Process the data
result <- advs %>%
  filter(PARAMCD == "WEIGHT") %>%  # Filter for PARAMCD = 'WEIGHT'
  group_by(TRTA, AVISIT) %>%       # Group by TRTA and AVISIT
  summarise(mean_AVAL = mean(AVAL, na.rm = TRUE)) %>%  # Calculate mean of AVAL
  ungroup()

# Print the result
print(result)
```



Confidential Information – Do Not Copy or Distribute

ChatGTP: Codes Development – SAS

Prompt :

Write a **SAS script** that groups data by “TRTA" and “AVISIT" columns in "advs.xpt"

and calculates the mean of “AVAL" column for each group at PARAMCD =‘WEIGHT’

Output :

```
libname mydata xport 'path_to_advs.xpt';  /* Replace with the actual path to advs.xpt */

data weight_data;
   set mydata.advs;
   if PARAMCD = 'WEIGHT'; /* Filter data for PARAMCD = 'WEIGHT' */
run;

proc sql;
   create table weight_means as
   select TRTA, AVISIT,
          mean(AVAL) as mean_weight
   from weight_data
   group by TRTA, AVISIT;
quit;

proc print data=weight_means;
run;
```

ChatGPT: Codes Development – Python

Prompt :

Write a **Python script** that groups data by “TRTA" and “AVISIT" columns in "advs.xpt"

and calculates the mean of “AVAL" column for each group at PARAMCD =‘WEIGHT’

Output :

```
import pandas as pd
import xport

# Load the .xpt file using xport
with open('advs.xpt', 'rb') as f:
    data = xport.load(f)

# Convert the xport data to a pandas DataFrame
df = pd.DataFrame(data['ADVS'])

# Filter the data for PARAMCD = 'WEIGHT'
df_weight = df[df['PARAMCD'] == 'WEIGHT']

# Group by TRTA and AVISIT, then calculate the mean of AVAL
grouped_data = df_weight.groupby(['TRTA', 'AVISIT'])['AVAL'].mean().reset_index()

# Output the grouped data
print(grouped_data)
```

ChatGPT: Codes Debugging – Python

Prompt :

Debug below <python codes> using <error message>.

<python codes>:

```
import pandas as pd
import xport

# Load the .xpt file using xport
with open('advs.xpt', 'rb') as f:
    data = xport.load(f)

# Convert the xport data to a pandas DataFrame
df = pd.DataFrame(data['ADVS'])
# Filter the data for PARAMCD = 'WEIGHT'
df_weight = df[df['PARAMCD'] == 'WEIGHT']

# Group by TRTA and AVISIT, then calculate the mean of AVAL
grouped_data = df_weight.groupby(['TRTA', 'AVISIT'])['AVAL'].mean().reset_index()

# Output the grouped data
print(grouped_data)
```

</python codes>

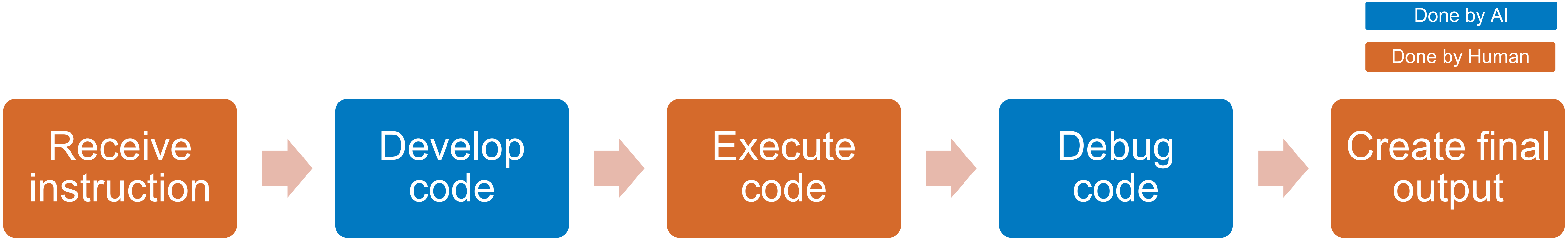
<error message>

```
TypeError: Could not convert ['aval'] to numeric
```

</error message>

Confidential Information – Do Not Copy or Distribute

Programming workflow using Vibe Coding - ChatGPT



Done by AI

Done by Human

Receive instruction

Develop code

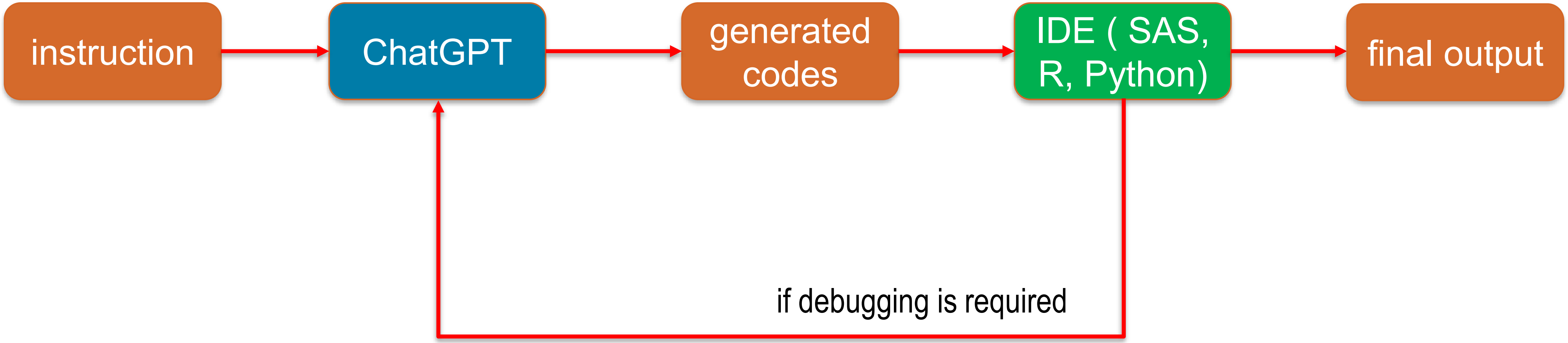
Execute code

Debug code

Create final output

Confidential Information – Do Not Copy or Distribute

Programming workflow using ChatGPT – Detailed



Question :

- Can we automate this workflow further?

AI Agent

- System that can interact with its environment, collect data, and autonomously peform tasks and make decisions to achieve the goals.

- Main Components
  - LLM
  - Prompts
  - Tools
  - Memory (short-term /long-term memory, RAG)

- Like Junior stat programmer who
  - has a knowledge of coding
  - with the instruction,
  - an access to tools (SAS, R, Python, functions & macros) and data
  - understands our workflow

AI Agentic workflow

AI Agentic Workflow is a workflow or process where an **AI agent** not only generates outputs (like code or text) but also **plans, executes, monitors, and adapts its actions** step by step to achieve a goal.

Ex. Like Junior Stat Programmer,
- Develop codes
- Execute codes
- Find the errors
- Debug codes
- Re-execute codes
- Do debugging and executing the codes until the codes are error-free or He or She can't resolve the errors anymore.
- Then, deliver the final results.

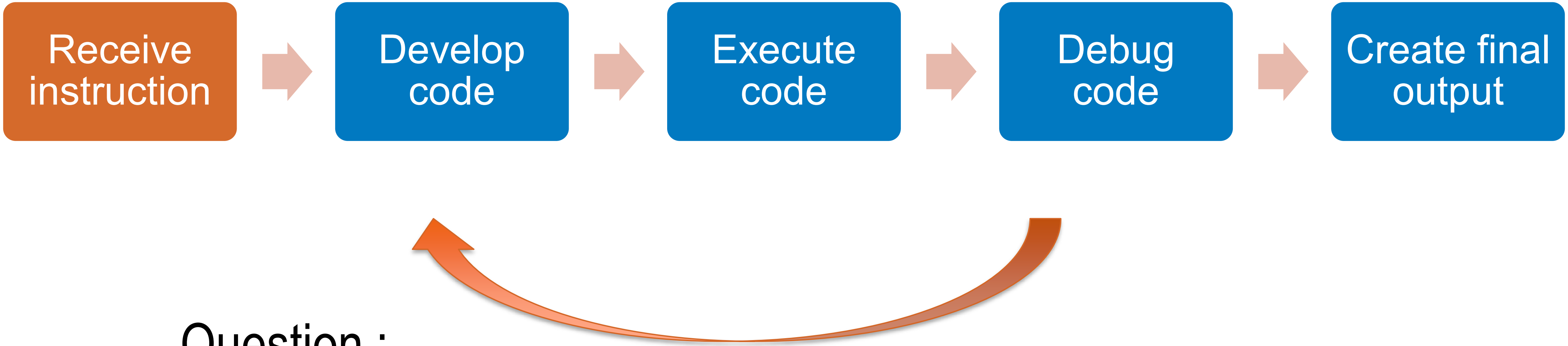
Clinvia

Confidential Information – Do Not Copy or Distribute

Programming workflow using Vibe Coding AI Agents

🟦 Done by AI

🟧 Done by Human



Question :

- What is the problem in this picture?

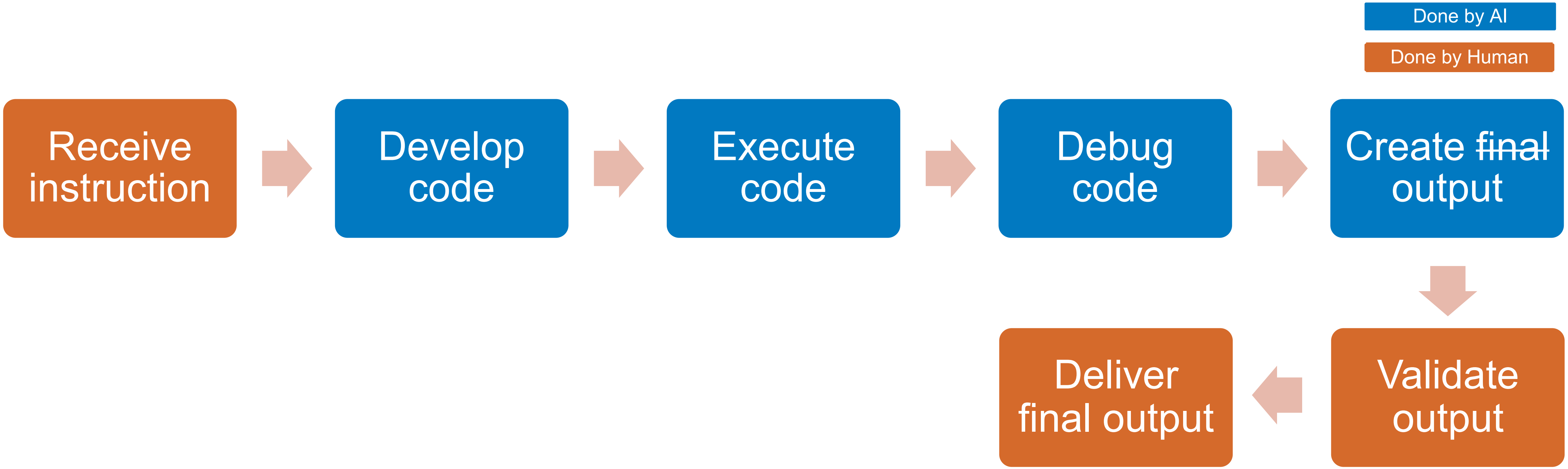
Confidential Information – Do Not Copy or Distribute

ChatGPT or AI Agent is like a very smart, capable, recently graduated intern/new hire.

=

What will we do with the results from an intern?

Programming workflow using Vibe Coding AI Agents with human involvement – AI Agentic workflow



Done by AI

Done by Human

Receive instruction

Develop code

Execute code

Debug code

Create ~~final~~ output

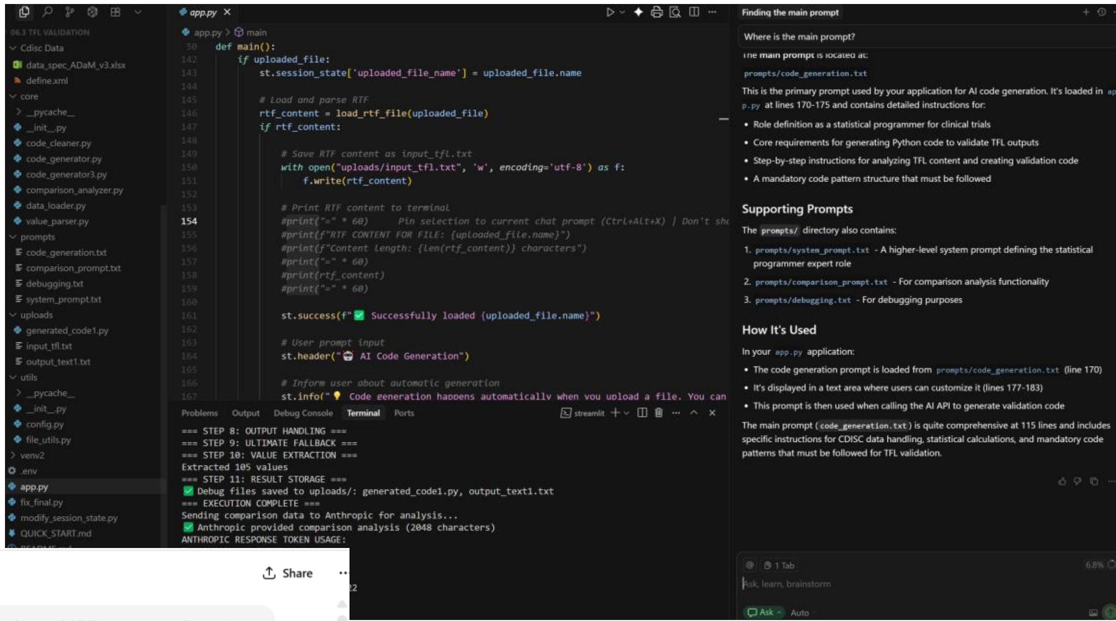
Validate output

Deliver final output

Confidential Information – Do Not Copy or Distribute

Vibe Coding AI Agents

- **ChatGPT / Claude / Gemini** – web-based prompt engines for ideation
- **Cursor** – AI-native IDE built around conversational programming
- **Windsurf** – context-aware code editing and real-time collaboration
- **GitHub Copilot** – inline suggestions in VS Code, RStudio, Jupyter



```python
def main():
    if uploaded_file:
        st.session_state['uploaded_file_name'] = uploaded_file.name

        # Load and parse RTF
        rtf_content = load_rtf_file(uploaded_file)
        if rtf_content:

            # Save RTF content as input_tfl.txt
            with open("uploads/input_tfl.txt", 'w', encoding='utf-8') as f:
                f.write(rtf_content)

            # Print RTF content to terminal
            #print("=" * 60)     Pin selection to current chat prompt (Ctrl+Alt+X) | Don't shc
            #print(f"RTF CONTENT FOR FILE: {uploaded_file.name}")
            #print(f"Content length: {len(rtf_content)} characters")
            #print("=" * 60)
            #print(rtf_content)
            #print("=" * 60)

            st.success(f"✅ Successfully loaded {uploaded_file.name}")

            # User prompt input
            st.header("🤖 AI Code Generation")

            # Inform user about automatic generation
            st.info("💡 Code generation happens automatically when you upload a file. You can
```

```
=== STEP 8: OUTPUT HANDLING ===
=== STEP 9: ULTIMATE FALLBACK ===
=== STEP 10: VALUE EXTRACTION ===
Extracted 105 values
=== STEP 11: RESULT STORAGE ===
✅ Debug files saved to uploads/: generated_code1.py, output_text1.txt
=== EXECUTION COMPLETE ===
Sending comparison data to Anthropic for analysis...
✅ Anthropic provided comparison analysis (2048 characters)
ANTHROPIC RESPONSE TOKEN USAGE:
```

Finding the main prompt

Where is the main prompt?

The **main prompt** is located at:

`prompts/code_generation.txt`

This is the primary prompt used by your application for AI code generation. It's loaded in `ap p.py` at lines 170-175 and contains detailed instructions for:

- Role definition as a statistical programmer for clinical trials
- Core requirements for generating Python code to validate TFL outputs
- Step-by-step instructions for analyzing TFL content and creating validation code
- A mandatory code pattern structure that must be followed

Supporting Prompts

The `prompts/` directory also contains:

1. `prompts/system_prompt.txt` - A higher-level system prompt defining the statistical programmer expert role
2. `prompts/comparison_prompt.txt` - For comparison analysis functionality
3. `prompts/debugging.txt` - For debugging purposes

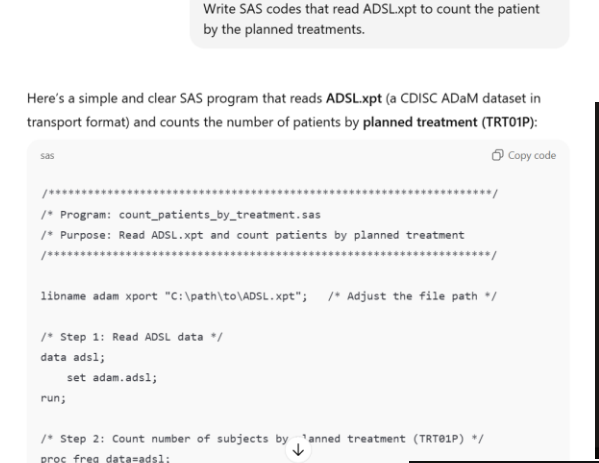
How It's Used

In your `app.py` application:

- The code generation prompt is loaded from `prompts/code_generation.txt` (line 170)
- It's displayed in a text area where users can customize it (lines 177-183)
- This prompt is then used when calling the AI API to generate validation code

The main prompt (`code_generation.txt`) is quite comprehensive at 115 lines and includes specific instructions for CDISC data handling, statistical calculations, and mandatory code patterns that must be followed for TFL validation.

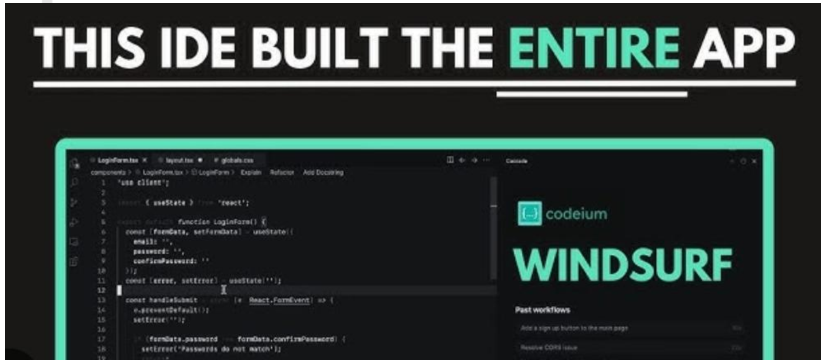
Ask, learn, brainstorm



Write SAS codes that read ADSL.xpt to count the patient by the planned treatments.

Here's a simple and clear SAS program that reads **ADSL.xpt** (a CDISC ADaM dataset in transport format) and counts the number of patients by **planned treatment (TRT01P)**:

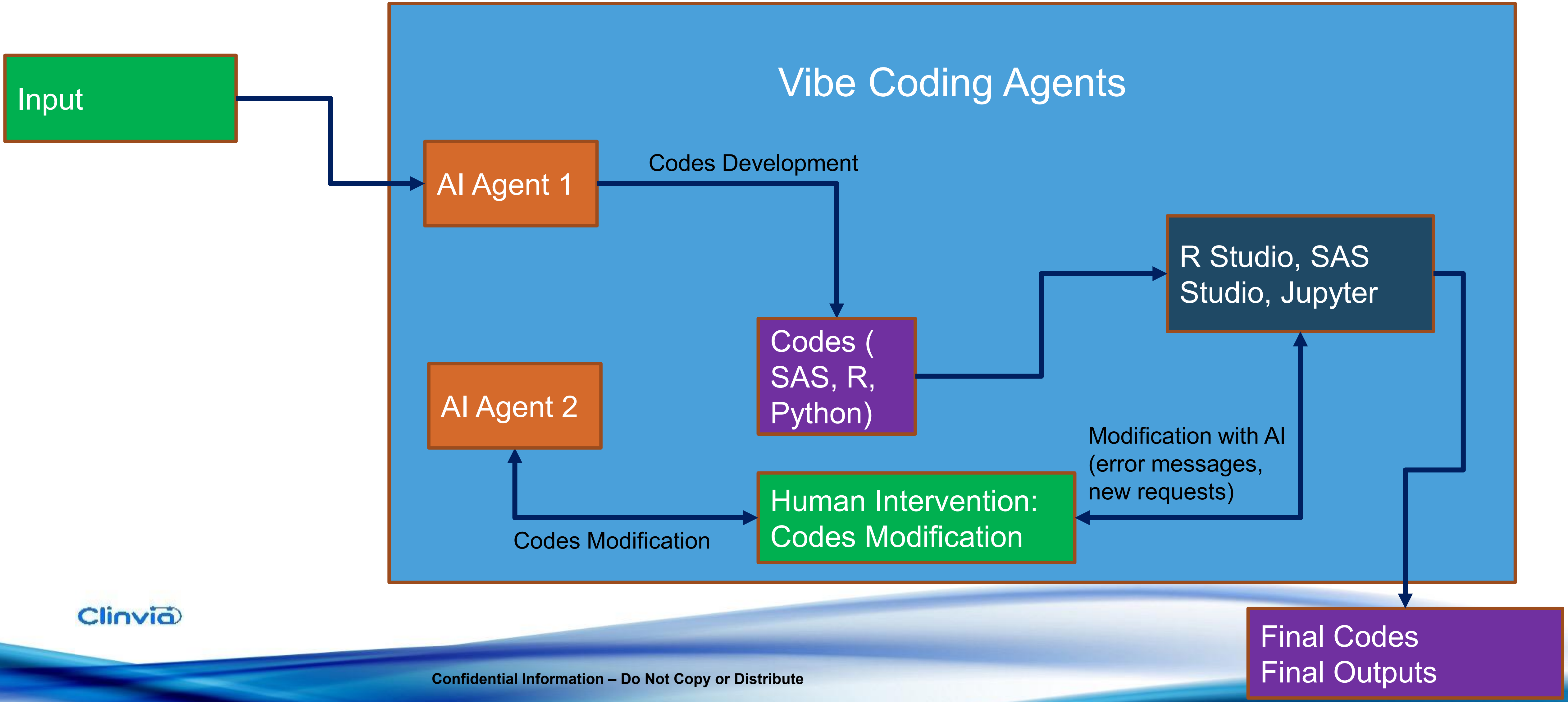
```sas
/**********************************************************************/
/* Program: count_patients_by_treatment.sas
/* Purpose: Read ADSL.xpt and count patients by planned treatment
/**********************************************************************/

libname adam xport "C:\path\to\ADSL.xpt";   /* Adjust the file path */

/* Step 1: Read ADSL data */
data adsl;
    set adam.adsl;
run;

/* Step 2: Count number of subjects by planned treatment (TRT01P) */
proc freq data=adsl;
    tables TRT01P / nocum nopercent;
```



Why customized Vibe Coding Agent for Biometrics works?

- Generic Coding AI agents lack domain knowledge
- Biometrics = highly specialized: CDISC, ADaM, TLFs, Compliance
- Custom agents embed clinical trial logic + GxP validation rules
- Data Privacy and Security
- Dirty Data

Vibe Coding Agents Architecture Example



Input

Vibe Coding Agents

AI Agent 1

Codes Development

R Studio, SAS Studio, Jupyter

Codes (SAS, R, Python)

AI Agent 2

Modification with AI (error messages, new requests)

Human Intervention: Codes Modification

Codes Modification

Final Codes
Final Outputs

Key Benefits on Vibe-coding

- Dramatic reduction in repetitive coding tasks
- Accelerates QC and development cycles
- Enables multi-language translation and consistency
- Democratizes programming across experience levels
- Improves documentation and traceability

Confidential Information – Do Not Copy or Distribute

Potential Risks on Vibe-Coding

- Hallucinated or Inaccurate code outputs
- Limited domain understanding
- Regulatory & Compliance challenges (audit trail, validation, Data Privacy)
- Risk of over-reliance on AI judgment

Confidential Information – Do Not Copy or Distribute

Governance & Validation framework

- **Regulatory Importance:** Vibe coding outputs must meet the same quality standards as traditionally produced results – traceability, reproducibility, accuracy, compliance.

- **Biometrics Leadership Role:** Must establish rigorous, risk-based validation process/systems tailored for Vibe-coding agentic workflow & process.

- **Framework Components:**
  - •Risk-based validation planning
  - •Verification of Vibe-coding workflow output accuracy
  - •Application of GxP / GAMP 5 principles
  - •Documented human approval and test evidence(e.g., plans, protocols, test scripts/results)
  - •Version control and audit trails
  - •Package validation

Future of Vibe-Coding and US

- Evolving Roles
- New Skill Sets Building
- AI and Human Collaboration
- Leading into the future

Confidential Information – Do Not Copy or Distribute

Evolving Roles

Now

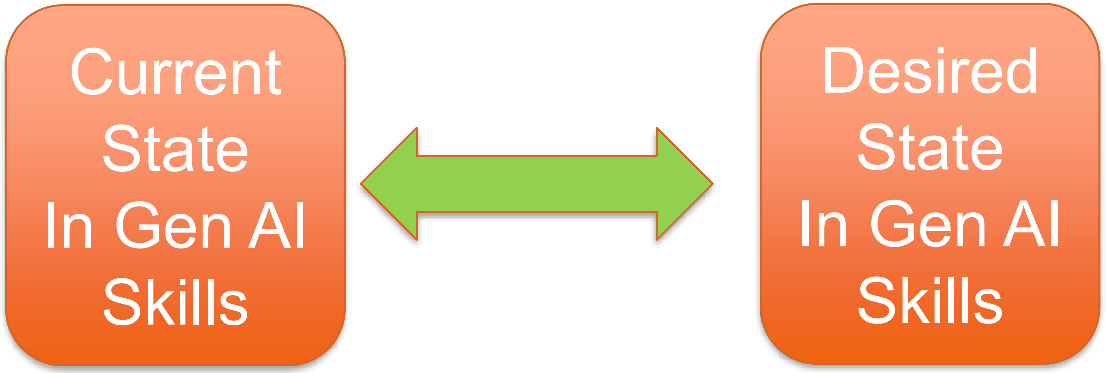
- Coder
- Syntax Expert
- Individual Work

Future

- AI Manager
- Validation Expert
- Collaborative AI augmented

Confidential Information – Do Not Copy or Distribute

New Skill Sets Building



Current State In Gen AI Skills

Desired State In Gen AI Skills

Assessment

- Existing skill gaps
- Implement targeted training initiatives.

- Focus areas for capability building:
    - Prompt engineering
    - Technical proficiency with Vibe Coding tools (e.g., ChatGPT, Copilot, Cursor)
    - Validation skills for AI-generated outputs
    - Understanding of Agentic workflow – AI and Human Collaboration Policy
    - More Humane Characteristics – EQ, Creativity, Critical Thinking, Collaboration, Communication

Confidential Information – Do Not Copy or Distribute

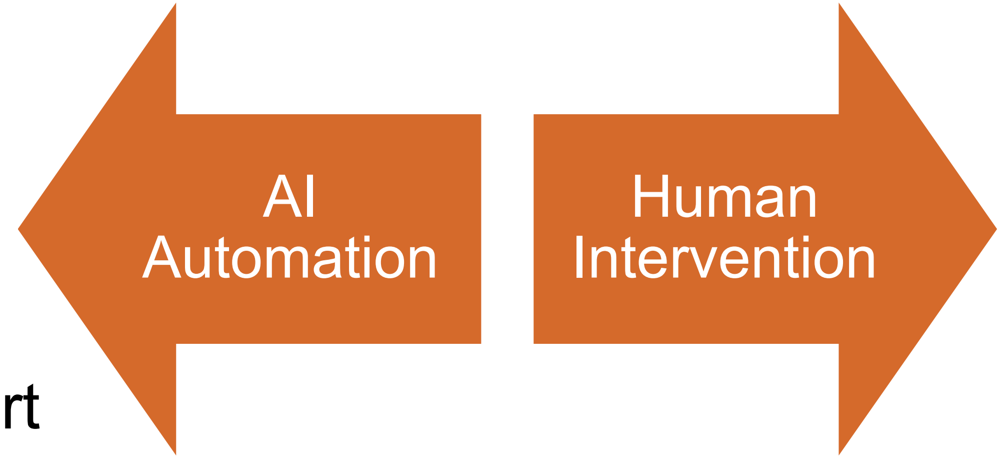
Human vs Vibe Coding Agents Collaboration Policy

- Clear delineation
  - AI-driven activities (Vibe Coding)
  - Human-driven activities

- AI roles
  - *Assistant*: e.g., generating initial code or report templates
  - *Autonomous Operator*: e.g., handling routine coding development and checks.

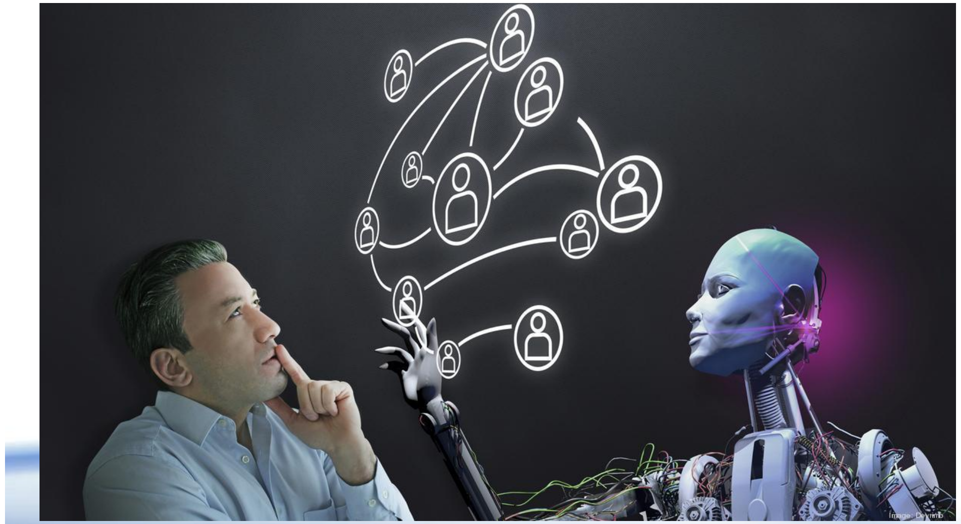
- Human intervention- required to ensure oversight and accountability.

- AI Validation and Audit Framework



AI Automation

Human Intervention



Leading into the future – Vibe Coding AI Strategy

- **Define AI-augmented Biometrics Roadmap using Vibe-coding.**

- **Position Biometrics as the innovation hub** within clinical development: the bridge between data, analytics, application and AI.

- Shift from "Submission deliverables" to "building intelligent AI powered team."

- Lower language barriers : from SAS to multi-lingual programming. Use the best tools for the right use cases.

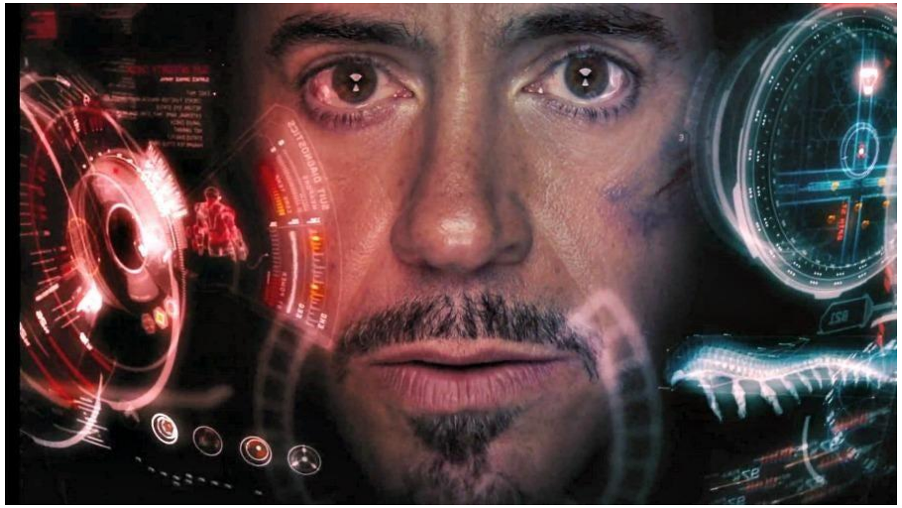
The impact of Vibe-Coding?

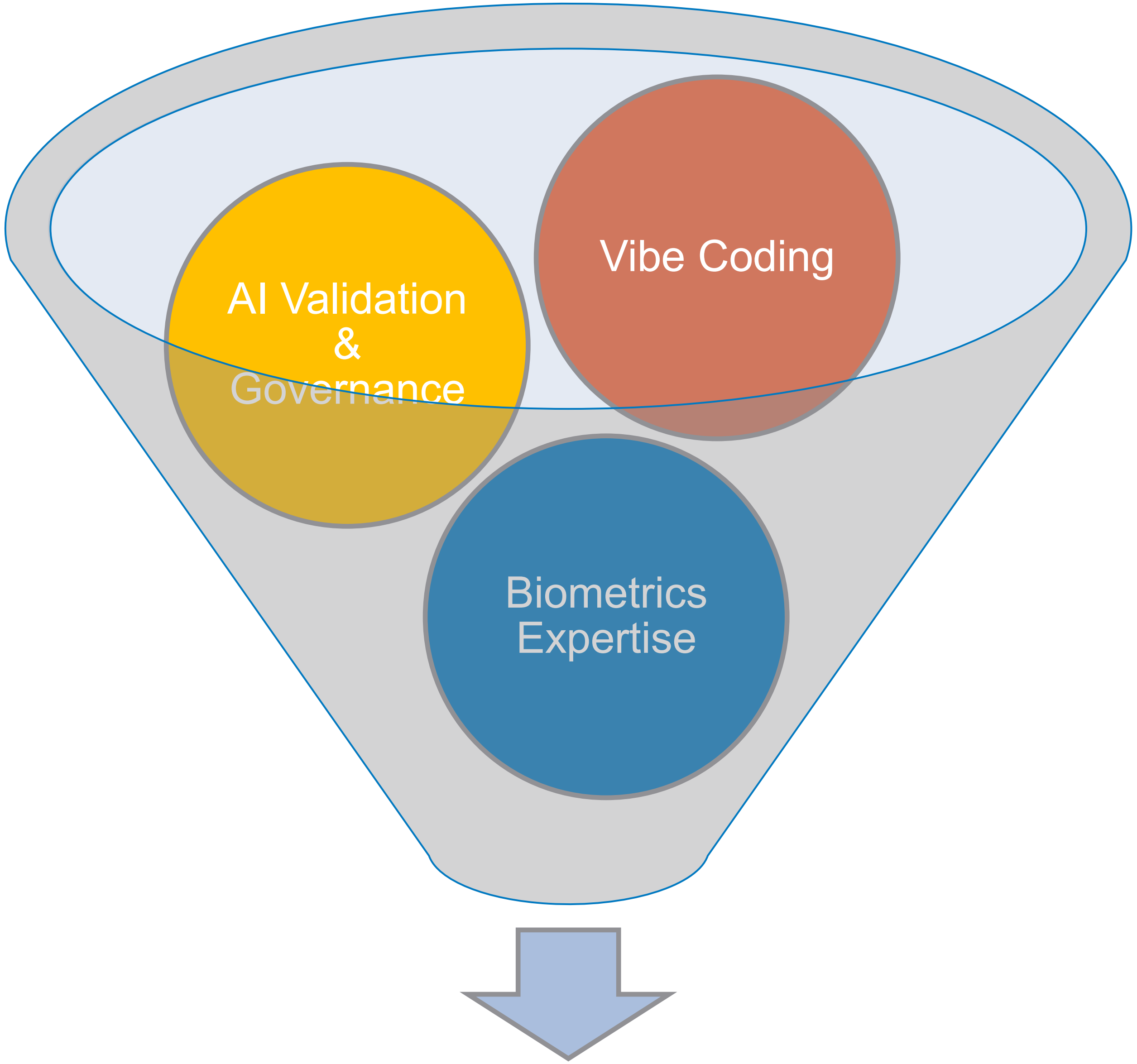
- Bain & Company : Average developer time savings: 10–15% across all engineering tasks.
- Github & Stack Overflow : Efficiency boosts for targeted tasks (boilerplate, documentation, testing), 30–60% time savings.
- Atlassian found that 99% of developers reported time savings using AI tools, with 68% estimating more than 10 hours saved per week.
- JPMorgan Chase measured up to 20% overall software engineering efficiency gain with AI.
- Federal Reserve Bank of St. Lous : Higher reported savings for daily AI users, who save up to 33% more time per coding hour by automating repetitive work.

Confidential Information – Do Not Copy or Distribute

Discussion :

Will Vibe-coding replace US or empower US as a programmer?





Vibe-coding powered “Super Biometrics Team (Avenger)”

Kevin Lee at kevin@clinvia.com

LinkedIn : https://www.linkedin.com/in/hellokevinlee

Appendix

Clinvia

Proprietary and Confidential Information – Do Not Copy or Distribute