



Revolutionize Your Analysis in Stata and R

AI Agent-Assisted Workflow with GitHub Copilot and Claude

Eduard Bukin ebukin@worldbank.org

Distributional Impact of Policies
Fiscal Policy and Growth Department

2026-02-05

Motivation

Chat and Web-Based AI tools are impressive!

[ChatGPT](#) | [Copilot](#) | [Gemini](#) | [WB MAI](#)

Use familiar technologies:

- Web-Browser,
- Chat,
- Stata editor,
- copy-paste ...

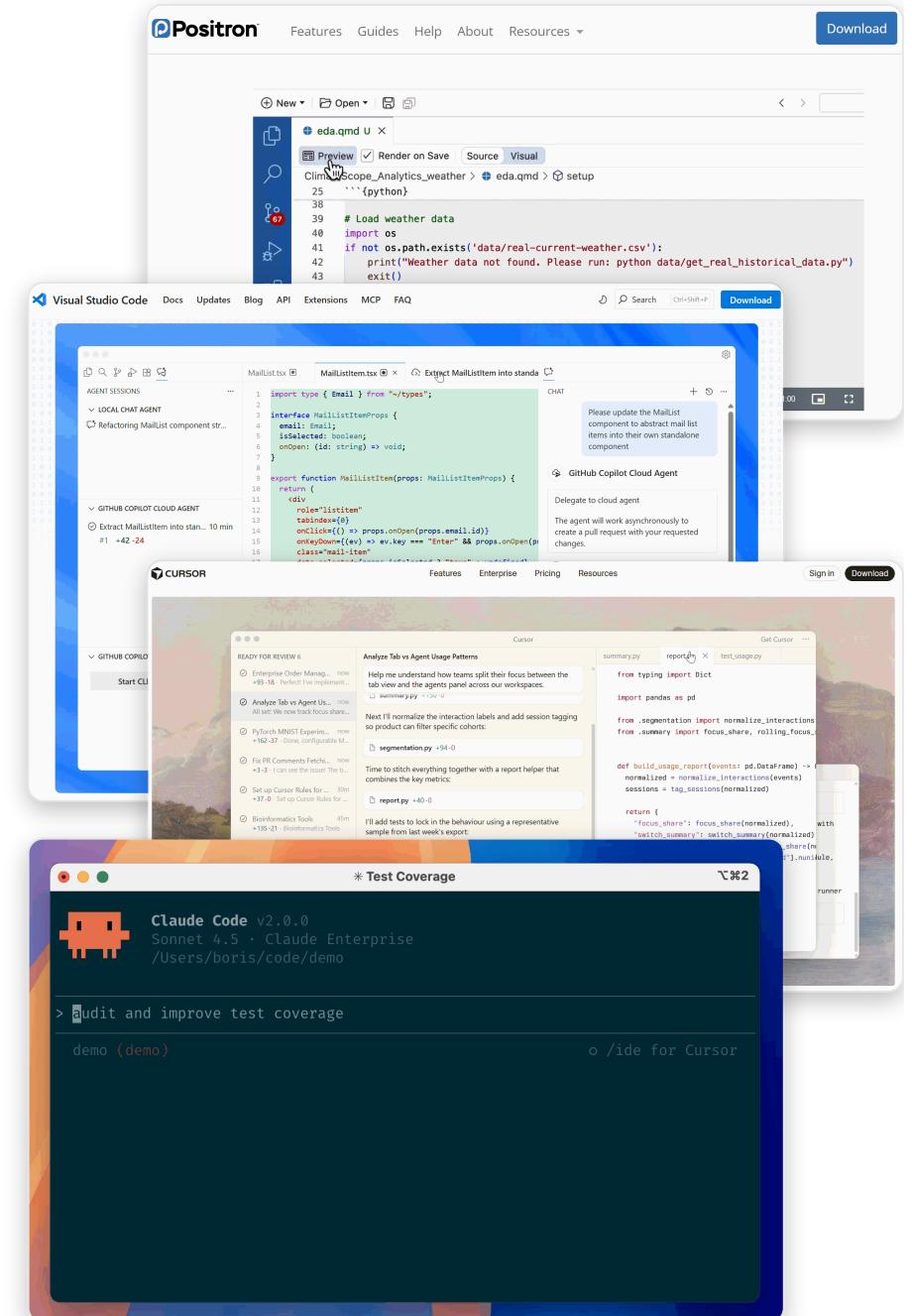
Is this the best way to use AI for data analysis ?

In fact, there are many AI-powered **Integrated Development Environments (IDEs)** for coding and data science!

The goal

of this seminar is to introduce you to AI-assisted data analysis with **Positron IDE** and **GitHub Copilot | Claude**.

There are many IDEs →



Agenda

- Introduce several AI-Concepts (vocabulary)
- Share experience of using AI-assisted workflow in Positron with Stata and R
- Provide kick-off instructions and resources.



WKPEP Seminar Series | Eduard Bukin | February 2026



Key Concepts

What do we need to know about modern analysis with AI?

1.  **AI Integrated IDE:** Chat | Agent | Inline Completion
2.  **Context awareness:** How AI understands your project
3.  **Model Context Protocol (MCP):** Universal adapter for AI
4.  **GitHub Copilot | Claude:** LLM providers
5.  **Efficient prompting:** Getting the best results
6.  **Caveats and limitations:** What to watch out for

AI Integrated IDE

The screenshot shows a Quarto document titled "weather_analysis_report.qmd" in the main editor area. The code is written in Python 3.9.6 (Global) and performs a seasonal temperature analysis. The code includes creating a facet boxplot for 12 US cities across four seasons. The variable explorer on the right shows data frames like "cities_info", "df", and "df_viz", and a series "seasonal_temp". The plot viewer displays a facet boxplot titled "Temperature Distribution by Season Across Cities" with axes for Temperature (°C) and Season (fall, spring, summer, winter). The plot shows box plots for 12 cities: Austin, Chicago, Columbus, Dallas, Fort Worth, Houston, Los Angeles, New York, Philadelphia, Phoenix, San Antonio, and San Diego.

```

weather_analysis_report.qmd -- weather_analytics
weather_analysis_report.qmd •
Preview Render on Save Source Visual Insert Code Cell ...
weather_analysis_report.qmd > Temperature Analysis > Seasonal Temperature Patterns > (code cell)
88 ## Temperature Analysis
124 ### Seasonal Temperature Patterns
128 ````{python}
129 #| fig-width: 16
130 #| fig-height: 12
131
132 # Create seasonal temperature boxplots
133 seasonal_temp = (
134     ggplot(df, aes(x="season", y="temperature_c", fill="season"))
135     + geom_boxplot(alpha=0.7)
136     + facet_wrap("city", scales="free_y")
137     + labs(
138         title="Temperature Distribution by Season Across Cities",
139         x="Season",
140         y="Temperature (°C)",
141         fill="Season"
142     )
143     + theme(
144         figure_size=(16, 12),
145         axis_text_x=element_text(angle=45, hjust=1)
146     )
147 )
148
149 seasonal_temp.show()
150
151

```

CHAT

Anthropic

What does my active plot show?

Positron Assistant

- Used 1 reference
- Viewed the active plot.

Your active plot shows temperature distributions by season across 12 US cities using box plots. Each city has its own panel, and the four seasons (fall, spring, summer, winter) are shown in different colors on the x-axis.

Key features of the plot:

- Format:** Faceted box plots with one panel per city
- Data:** Temperature in Celsius on the y-axis, seasons on the x-axis
- Color coding:** Fall (coral/red), Spring (green), Summer (blue), Winter (purple)
- Geographic coverage:** Cities from Austin to San Diego, representing different climate zones

The plot reveals clear seasonal patterns:

- Phoenix shows the highest summer temperatures (reaching ~40°C)
- Northern cities (Chicago, New York, Philadelphia) have the greatest temperature ranges between seasons
- California cities (Los Angeles, San Diego) show the most moderate seasonal variation
- Winter temperatures vary dramatically by location, from below 0°C in northern cities to 10-15°C in southern/western cities

CONSOLE

TERMINAL

PROBLEMS 1

OUTPUT

PORTS

DEBUG CONSOLE

~demobot/weather_analytics

Python 3.9.6 ...

R 4.4.3

Add Context...

weather_analysis_report.qmd Current file

Python 3.9.6 (Global) Console session

Ask Assistant

Ask Claude 4 Sonnet

1 0 Quarto: 1.4.551

SESSION CONNECTIONS HELP VIEWER

VARIABLES

Python 3.9.6 (Global)

DATA

VALUES

PLOTS

Temperature Distribution by Season Across Cities

Austin Chicago Columbus Dallas

Fort Worth Houston Los Angeles New York

Philadelphia Phoenix San Antonio San Diego

Season

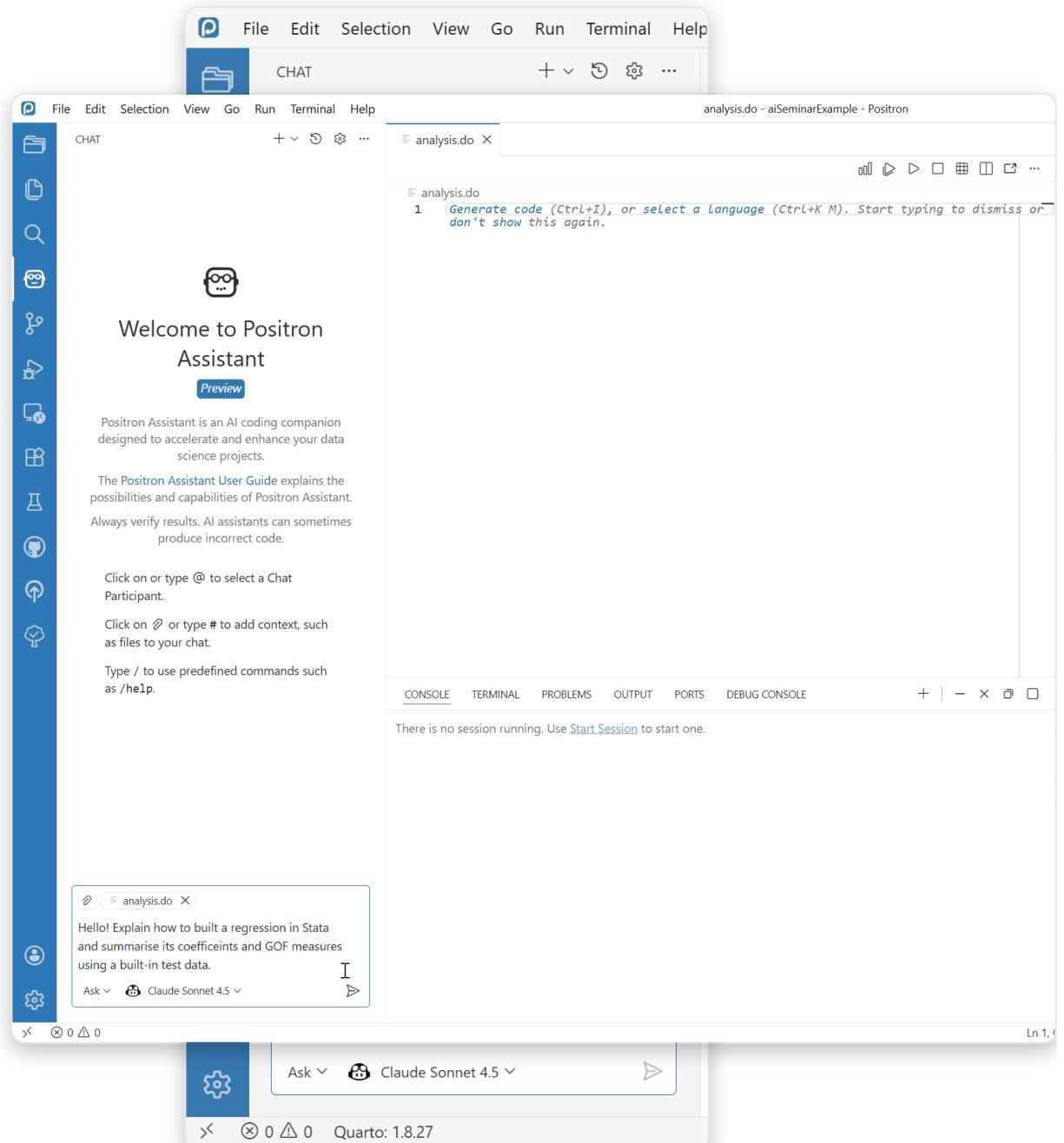
Temperature (°C)

fall spring summer winter

AI: Chat

Positron Assistant

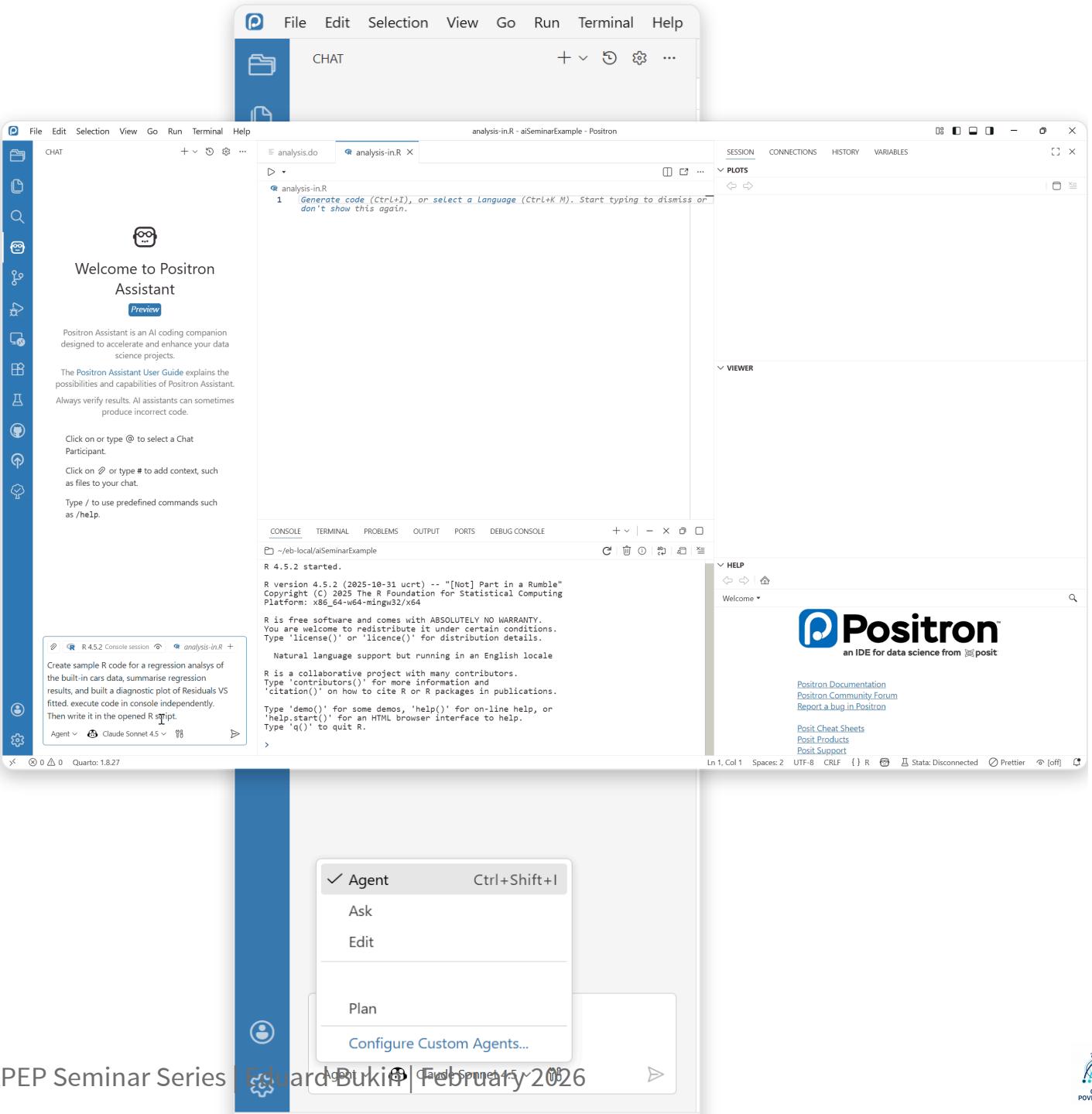
- Ask AI (Claude 4.5) through Github Copilot
 - Provides explanations, suggestions, and code snippets.
 - Integrates with project context, and code.
 - Learn more:
- [Assistant Chat](#)



AI: Agent

Positron Assistant

- Executes instructions.
- Acts independently
 - Runs code
 - Fixes errors
 - Learns
 - Reasons
- See more in the live demo!





AI: Inline Completion

Positron Inline Code Completion: Suggests code snippets as you type.

The screenshot shows the Positron IDE interface. At the top, there are two tabs: "analysis.do" (active) and "analysis-in.R". Below the tabs is a toolbar with various icons. The main area displays a Stata script named "analysis.do" with the following content:

```
analysis.do
1 // Load the built-in auto dataset
2 sysuse auto, clear
3
4 // Run a linear regression
5 // Dependent variable: price
6 // Independent variables: mpg, weight, foreign
7 regress price mpg weight foreign
8
9 // Display the regression results with robust standard errors
```

A cursor is positioned at the end of line 9. A vertical list of code suggestions is displayed on the right side of the editor window, starting with "I". At the bottom of the window, there are tabs for "CONSOLE", "TERMINAL", "OUTPUT", and "...". To the right of these tabs is a "Filter" input field and a dropdown menu set to "Stata". On the far right, there are additional interface controls.



Context Awareness

Positron accesses project metadata.

Why does it matter?

Thus AI ‘knows’:

- **Files str.:** Code, docs
- **Data:** Var. names, types
- **History:** Edits, commands
- **Environment:** Packages
- **Intent:** Current task
- **Results:** Output, errors

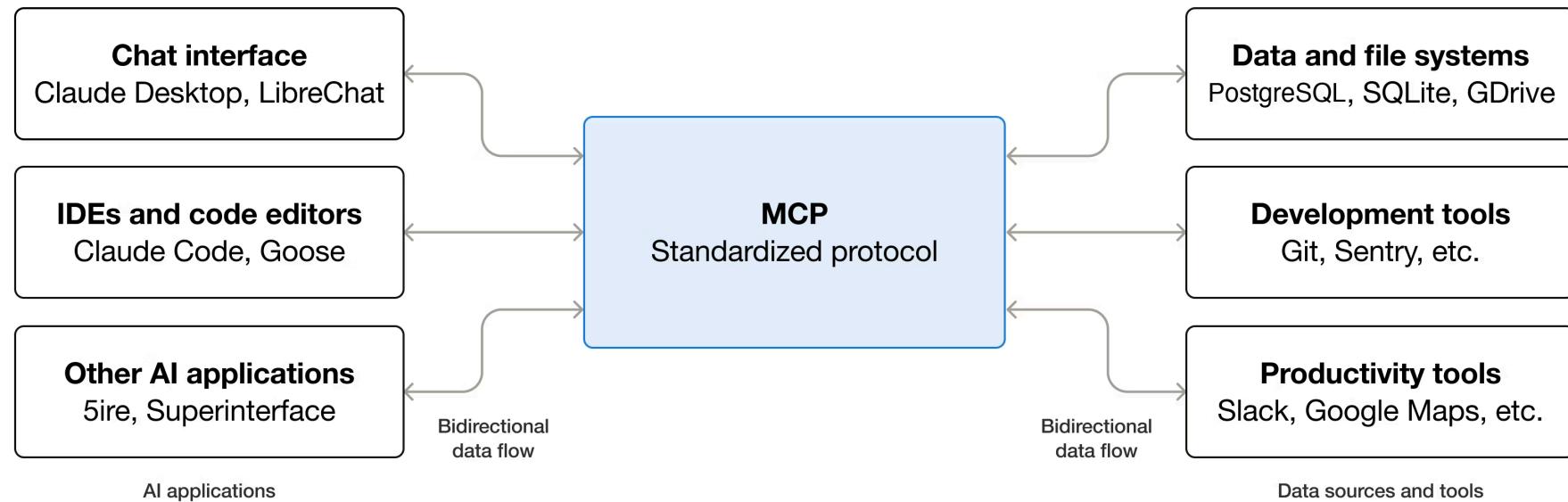
- Project-specific suggestions
- Understands dependencies
- Reduces hallucinations
- Improves efficiency



Model Context Protocol (MCP)

MCP is a universal adapter for AI—Anthropic— that connects **data flows**:

- [Stata-MCP](#) or [R MCP](#) for Stata/R → Positron → Copilot → LLMs.





GitHub Copilot | Anthropic Claude

GitHub Copilot

- WB-approved
- github.com/worldbank
- Uses **Claude | GPT**
- Integrates with IDEs
- github@worldbank.org
- **AI @ WB:** ai.worldbank.org

Choose your LLM:

- Claude Sonnet/Haiku/Opus
- OpenAI GPT-4/o1...

	Claude Sonnet 4.5	Claude Opus 4.1	Claude Sonnet 4	GPT-5	Gemini 2.5 Pro
Agentic coding <i>SWE-bench Verified</i>	77.2% 82.0% with parallel test-time compute	74.5% 79.4% with parallel test-time compute	72.7% 80.2% with parallel test-time compute	72.8% GPT-5	67.2%
Agentic terminal coding <i>Terminal-Bench</i>	50.0%	46.5%	36.4%	43.8%	25.3%
Agentic tool use <i>t2-bench</i>	Retail 86.2% Airline 70.0% Telecom 98.0%	Retail 86.8% Airline 63.0% Telecom 71.5%	Retail 83.8% Airline 63.0% Telecom 49.6%	Retail 81.1% Airline 62.6% Telecom 96.7%	—
Computer use <i>OSWorld</i>	61.4%	44.4%	42.2%	—	—
High school math competition <i>AIME 2025</i>	100% (python) 87.0% (no tools)	78.0%	70.5%	99.6% (python) 94.6% (no tools)	88.0%
Graduate-level reasoning <i>GPQA Diamond</i>	83.4%	81.0%	76.1%	85.7%	86.4%
Multilingual Q&A <i>MMMLU</i>	89.1%	89.5%	86.5%	89.4%	—
Visual reasoning <i>MMMU (validation)</i>	77.8%	77.1%	74.4%	84.2%	82.0%
Financial analysis <i>Finance Agent</i>	55.3%	50.9%	44.5%	46.9%	29.4%



Efficient Prompting

- **Be specific:**

“Write a Stata do-file to ...” / “Refactor this R function to ...”

- **Provide context:**

“Goal: X; Dataset: Y variables; Constraints: Z (WB rules, packages, runtime)”

- **Define expected output:**

“Save as regression_results.xlsx, format as APA table” / “Create bar chart with 95% CIs”

- **Summarize + clarify first:**

“Restate and ask clarifying questions before implementing”, “Explain why ...”, “Give alternatives with trade-offs...”

- **Iterate in small steps:**

“minimal changes”, “refine”

- **Set boundaries:**

“Don’t use ... data”, “Don’t print secrets, ask if in doubt.”, “Don’t change files”.



Limitations and Remedies

- **⚠ Wrong-but-plausible outputs / hallucinations:** code runs but logic is wrong
 - ✓ **Verify and validate:** ask the model to **explain and justify** the solution
- **⚠ Context limits:** not all files/data are in context; too large projects.
 - ✓ **Be explicit:** state assumptions, expected inputs/outputs, and references
- **⚠ Outdated knowledge:** suggested APIs/packages/options may have changed
 - ✓ **Teach the model:** provide references/links; ask it to learn
- **⚠ Over-reliance:** erodes fundamentals; mistakes slip through unchallenged
 - ✓ **Keep learning:** ask for step-by-step reasoning; request alternatives and trade-offs
- **⚠ Confidentiality / security / privacy**
 - ✓ **Constrain context:** exclude sensitive data; use `.copilot-ignore`; [AI @ WB](#)
- **⚠ Reproducibility:** answers can vary across sessions/models/settings
 - ✓ **Cutomize agents:** save prompts, use Git; create AI agents

Summary

Why use IDEs, not a web-browser-based workflow?

- Context-awareness
- Streamlined workflow
- Reduced friction

Why Positron?

- Built for data science, not software development
- Integrates with Stata, R, and Python seamlessly
- Advanced AI features for data analysis

Where to Start?

- Setup the software: follow [instructions](#)
- Reproduce demos:
 - [Live demo in Positron with Stata](#)
 - [Posit Conf 2025 Positron assistant Demo in R and Python](#)
- Ask AI's help to learn.
- Step out of your comfort zone:
 - learn and experiment with new technologies: [Git](#), [GitHub](#), [R](#), [Python](#)—they are there for a good reason!
- See more slides with [additional materials](#) below.

The screenshot shows a search results page from the World Bank Group website. The top navigation bar includes the World Bank Group logo and a search icon. Below the header, a breadcrumb trail shows 'GeoPov BBL Series > Revolutionizing Coding in Stata/R > Setup instructions'. The main content area features a large title 'Positron + Github Copilot: Setup Instructions' in bold black text. A horizontal scrollbar is visible at the bottom of the content area.

Live Demo

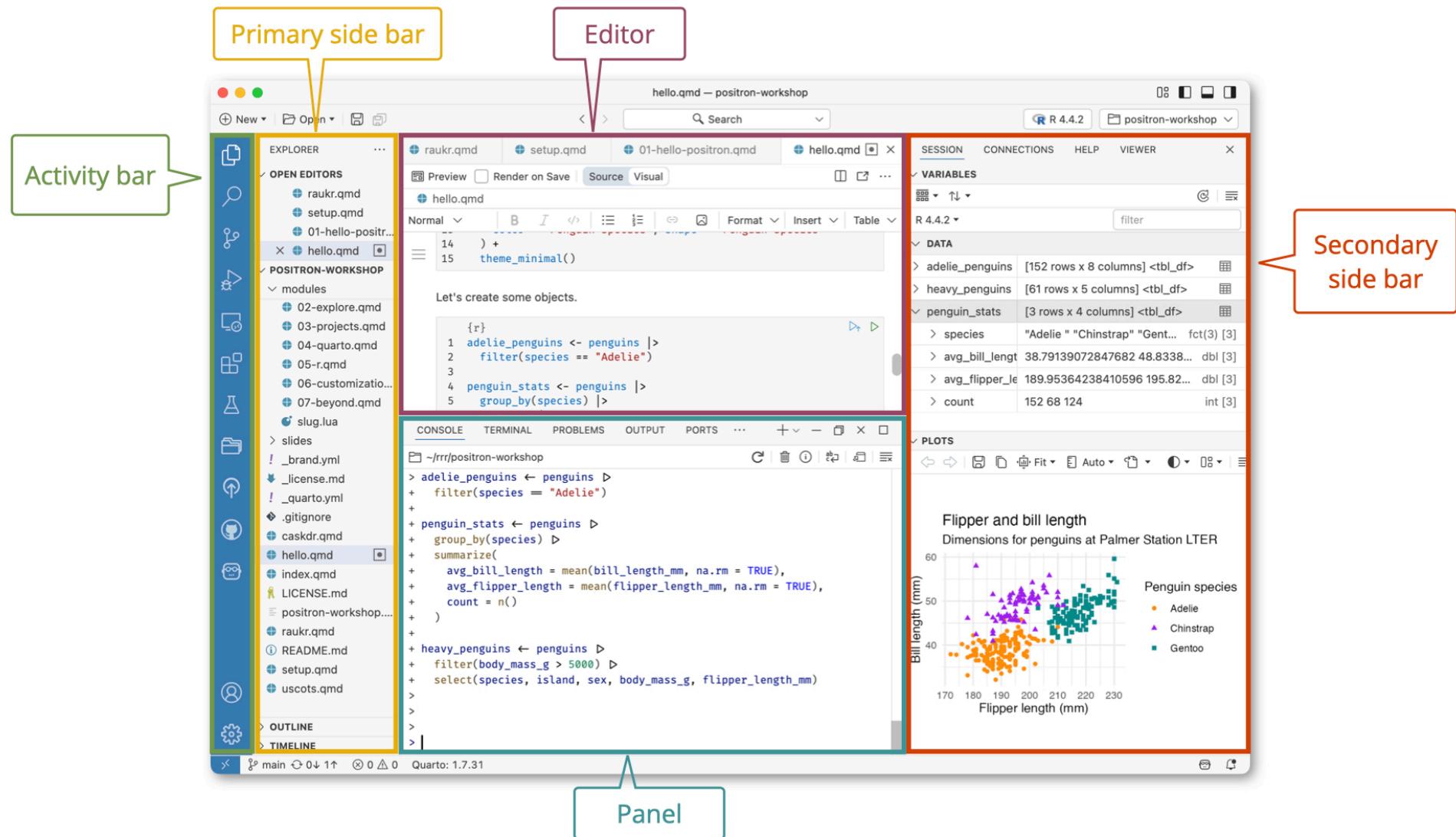
From an old analysis in Stata to an upgraded Stata+R reproducibility package in under 10 minutes!

- Download materials here:
github.com/WBGGeoPov/seminar-coding-with-ai-demo
- Watch the seminar recording:



Ask AI for help: “How do I download a project from GitHub and open it in Positron IDE? The link is: ...”

Live Demo: Positron IDE overview



Thank You! Questions?



Resources & Source Code

- Internal seminar page
- 🎥 Recording
- Positron + Stata: Live Demo (code)
- Slides and Supporting Materials
 - Slides
 - Setup Instructions
- Presentation Source Code
- GeoPov learning resources

Contact: Distributional Impact of Policies poverty@worldbank.org

Additional materials

Software Setup Overview



Full details: [Setup Instructions](#)

1. Install prerequisite software (via WB Software Center)

- Stata 19+, R 4.5+, Python 3.13+, Quarto, Git
- Install Python `uv` package: `pip install uv`

2. Install Positron IDE (system-level install): Request help from IT if needed

3. Install key extensions in Positron

- Stata MCP, Quarto

4. Connect GitHub and configure Positron Assistant

5. Start experimenting!

- Open assistant: `Ctrl+Shift+P` > “Ask Positron Assistant”
- Try: chat, agent mode, inline code completion

Positron IDE: Self-learning

[Positron](#): Modern AI-native
IDE for data science

- Built by [Posit](#) (creators of RStudio)
- Supports Stata, R, Python, and others
- Watch the [introduction video](#) →
- Try it yourself with examples in R and Python:
 - github.com/posit-dev/posit-conf-2025-positron-assistant-demo

Positron: The First Five Minutes (Isabella Velásquez, Posit) | posit::con...



Positron: Assistant

The screenshot shows the Positron IDE interface with the following components:

- Left Sidebar:** Includes icons for Chat (1), Anthropic, Positron Assistant (Used 1 reference), Viewed the active plot, Key features of the plot, and Plot reveals clear seasonal patterns.
- Code Editor:** A Quarto document titled "weather_analysis_report.qmd" under "weather_analytics". The code cell contains Python 3.9.6 (Global) code for "Seasonal Temperature Patterns".
- Variables Panel:** Shows variables like cities_info, df, df_viz, null_counts, season_counts, col, count, and seasonal_temp.
- Plots Panel:** Displays a facet boxplot titled "Temperature Distribution by Season Across Cities" for 12 US cities (Austin, Chicago, Columbus, Dallas, Fort Worth, Houston, Los Angeles, New York, Philadelphia, Phoenix, San Antonio, San Diego). The y-axis is Temperature (°C) ranging from -10 to 30, and the x-axis is Season (fall, spring, summer, winter).
- Bottom Status Bar:** Shows the file path (~demobot/weather_analytics), Python 3.9.6 (Global) and R 4.4.3 consoles, and the current line (Ln 149, Col 21).

Positron: Data explorer

The screenshot shows the Positron IDE interface for a Jupyter Notebook named `diamonds.qmd`. The interface includes:

- EXPLORER** panel on the left showing files like `.venv`, `diamonds_files`, `database.db`, `diamonds_analysis.qmd`, `diamonds_sqlite.R`, `diamonds.csv`, `diamonds.html`, `diamonds.qmd` (selected), `new-analysis.qmd`, `nyflights13.sqlite`, and `read_diamonds.py`.
- Code Cell** panel showing Python code for loading the diamonds dataset and creating a scatter plot.
- VARIABLES** panel showing the `diamonds` DataFrame and its columns: carat, cut, color, clarity, depth, table, price, x, y, z.
- PLOTS** panel displaying a scatter plot titled "Diamond Price vs Carat" with Price on the y-axis and Carat on the x-axis, colored by Cut (Ideal, Premium, Very Good, Good, Fair).
- CONSOLE** and **TERMINAL** panels at the bottom showing the executed code and its output.

Positron + Stata

1. Make sure **prerequisite software is installed** (Stata, R, Positron)
2. **Install Python** after that the **uv** package: `pip install uv`
3. **Install Stata MCP** in Positron and configure Stata path and Edition
4. Create a new Stata do-file, write some code, save it and press run it.