

Tips & Tricks from the maintainers of Positron



Sharon Wang & Melissa Barca

September 17, 2025

posit
conf(2025)





Today, we've got tips & tricks for...



Personalizing your coding space



Making data exploration more pleasant



Thoughtful pairing with Positron Assistant



Personalizing your coding space

Set up the essentials

Figure out your comfy layout

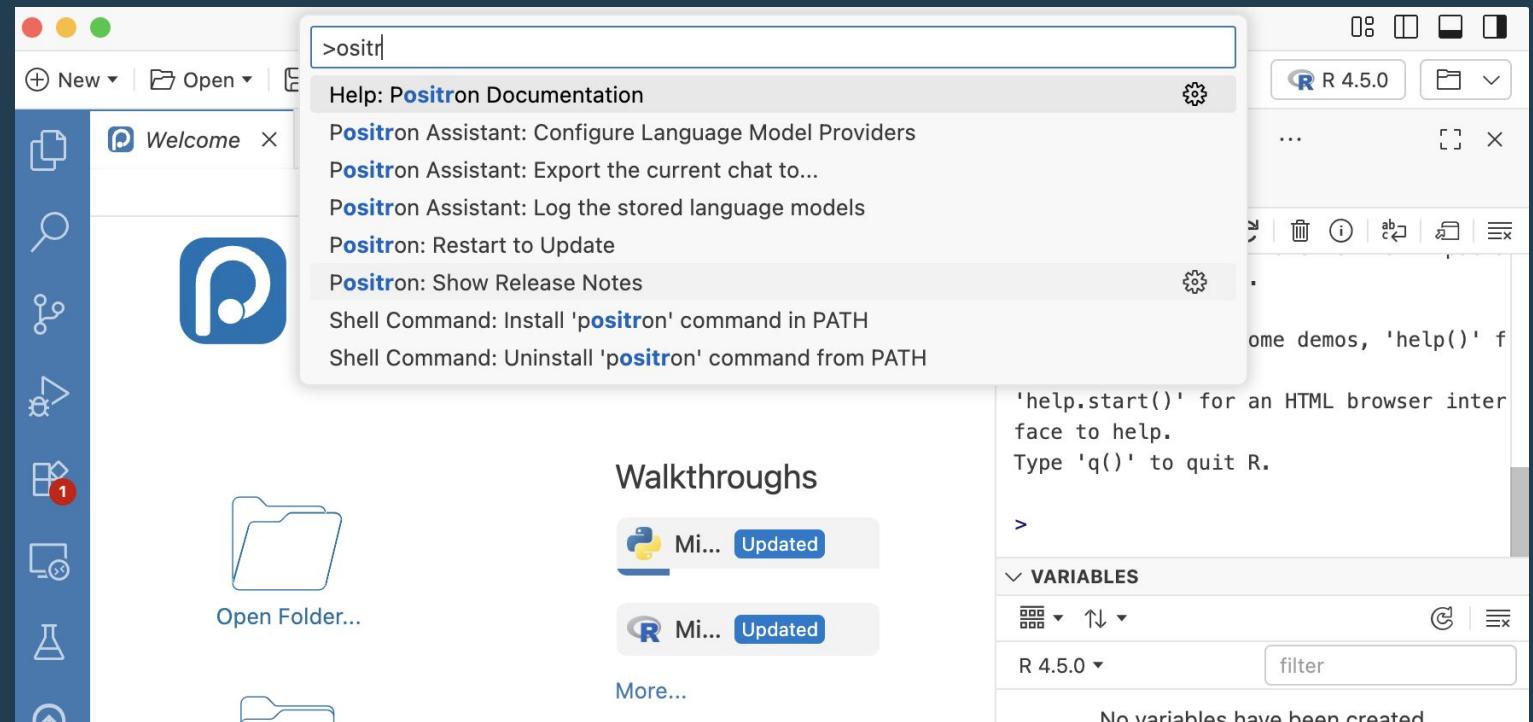
posit
conf (2025)



Command Palette

- Run any command, install extensions, toggle settings
- Great for discovering what's possible

Press Ctrl+Shift+P / Cmd+Shift+P to open



Walkthroughs

- Run command `Welcome: Open Walkthrough...`

```
>welcome open
```

Welcome: Open Walkthrough...



Walkthroughs

- Run command `Welcome: Open Walkthrough...`

```
>welcome open
```

[Welcome: Open](#) Walkthrough...

- Then type “migrating” and select a walkthrough

```
migrating
```

[Migrating from VSCode to Positron Python](#)

Learn how to get started with Python in Positron as a VSCode user!

[Migrating from RStudio to Positron R](#)

Learn how to get started with R in Positron as an RStudio user!

Walkthroughs

- VS Code to Positron
- RStudio to Positron

Migrating from VSCode to Positron

Learn how to get started with Python in Positron as a VSCode user!

The screenshot shows a guide titled "Migrating from VSCode to Positron". On the left, there's a sidebar with several items:

- Get to know the Positron UI
- Find your way around Positron
- Customize your workspace
- Select a Python interpreter
- Search built-in environments
- Customize your code editor
- Configure your command palette

At the bottom of the sidebar, there's a "Mark Done" button.

The main content area has a heading "Migrating from VSCode to Positron" and a sub-section "Get started with Python in Positron". Below that, there's a "Keyboard shortcuts" section with a "Enable RStudio keybindings" button, and other sections like "Projects, folders, and workspaces" and "Formatting your Python code". At the bottom, there's a "Mark Done" button.

Positron has multiple panes, built to support the way data scientists and Python developers work:

- Run Python code interactively in the [Console](#).
- The [Variables pane](#) lets you inspect and explore Python variables in your current session.
- The [Plots pane](#) displays plots created with libraries such as `matplotlib`, `seaborn`, or `plotnine`.
- The [Help pane](#) provides documentation lookups for Python objects and packages.
- Manage database connections and preview tables in the [Connections pane](#).

Many of the keybindings you are familiar with from RStudio are available by default in Positron. For example, you can:

- Use `Cmd/Ctrl + Enter` to run the current R statement (or selection) in the Console.
- Restart R with `Cmd/Ctrl + Shift + 0`.
- Insert a pipe with `Cmd/Ctrl + Shift + M`.

Some RStudio keybindings are not available by default, but you can [opt in](#) to them using a [setting](#). If you opt in, you can then:

- Focus the Source Editor with `Cmd/Ctrl + 1` and focus the Console with `Cmd/Ctrl + 2`.
- Comment or uncomment the current line with `Cmd/Ctrl + Shift + C`.
- Create a new R file with `Cmd/Ctrl + Shift + N`.

Opting in to this setting will overwrite some of Positron's default keybindings, so you might then also prefer to change or remove some of these newly enabled keybindings. Learn how to customize keyboard shortcuts using the [documentation from VS Code](#).

Layout

- Looking for a four pane data science layout?

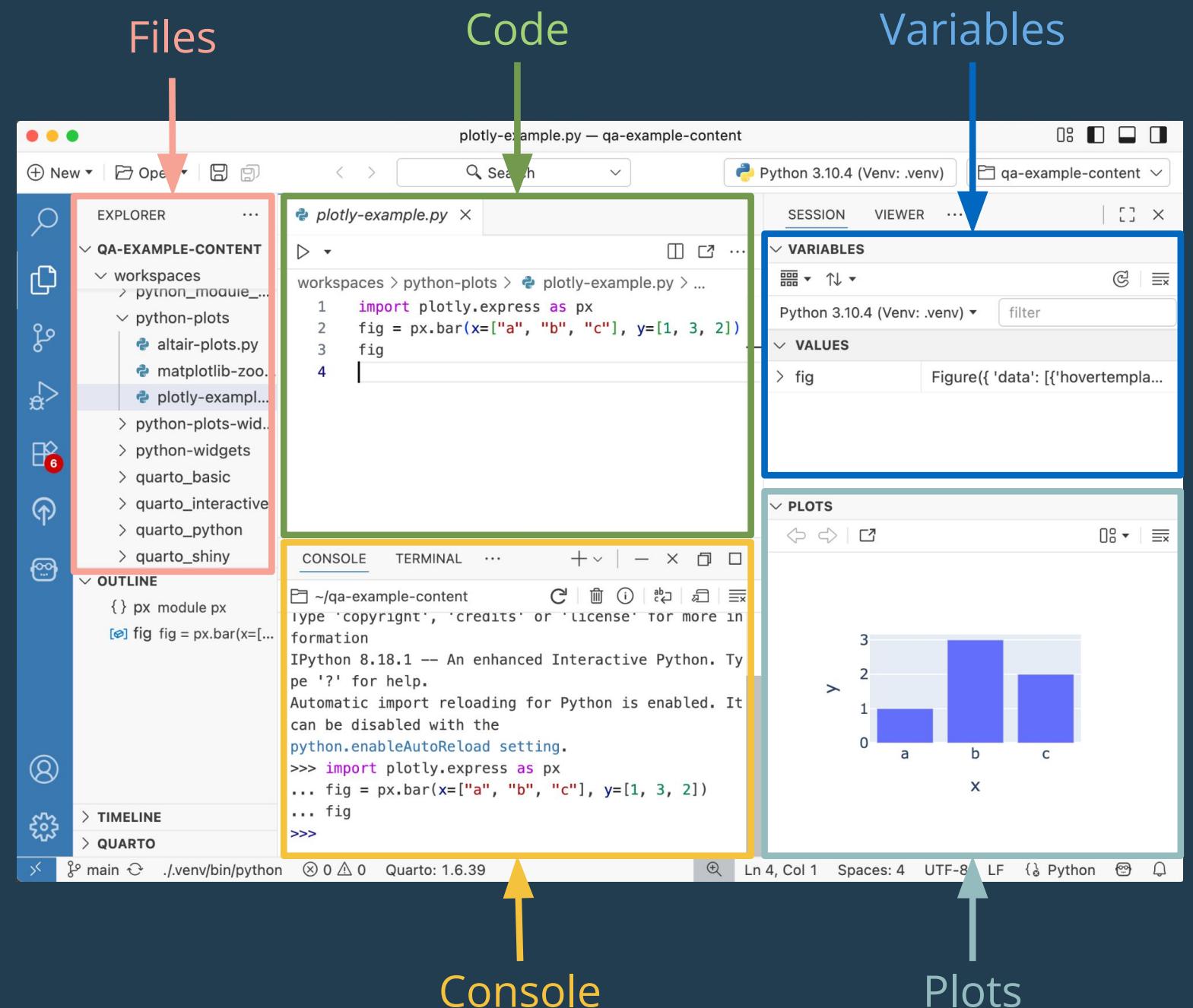
Run command **View: Stacked Layout**

```
>view stacked
```

View: Stacked Layout

Layout

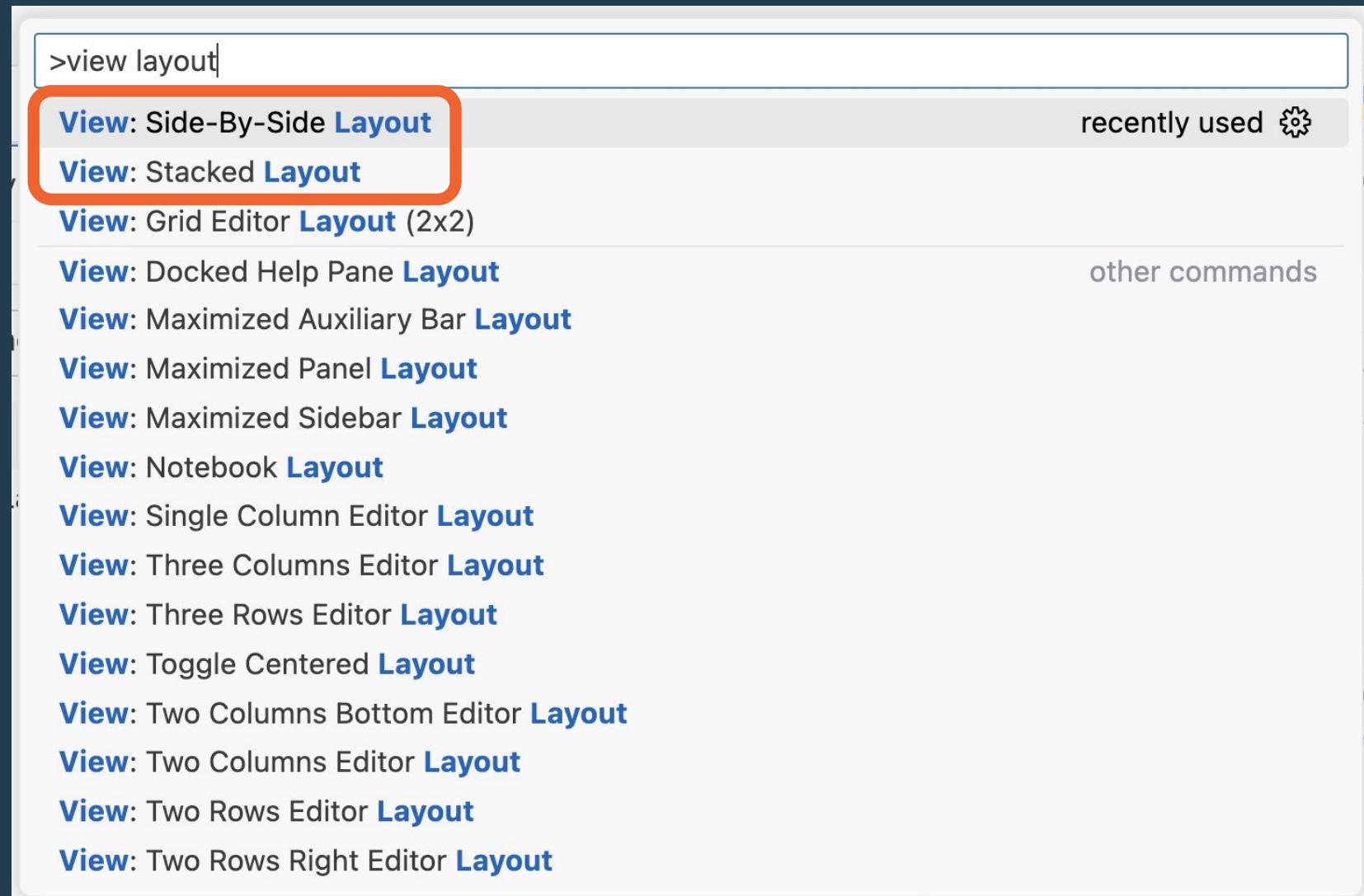
- Looking for a four pane data science layout?
- Run command View: Stacked Layout



Layout

- For even more customization...

Run command **Customize Layout...** or search for **view layout**



Interpreter Picker

- Indicates the active Console session and helps us populate the Variables

The screenshot shows a Jupyter Notebook interface with the following details:

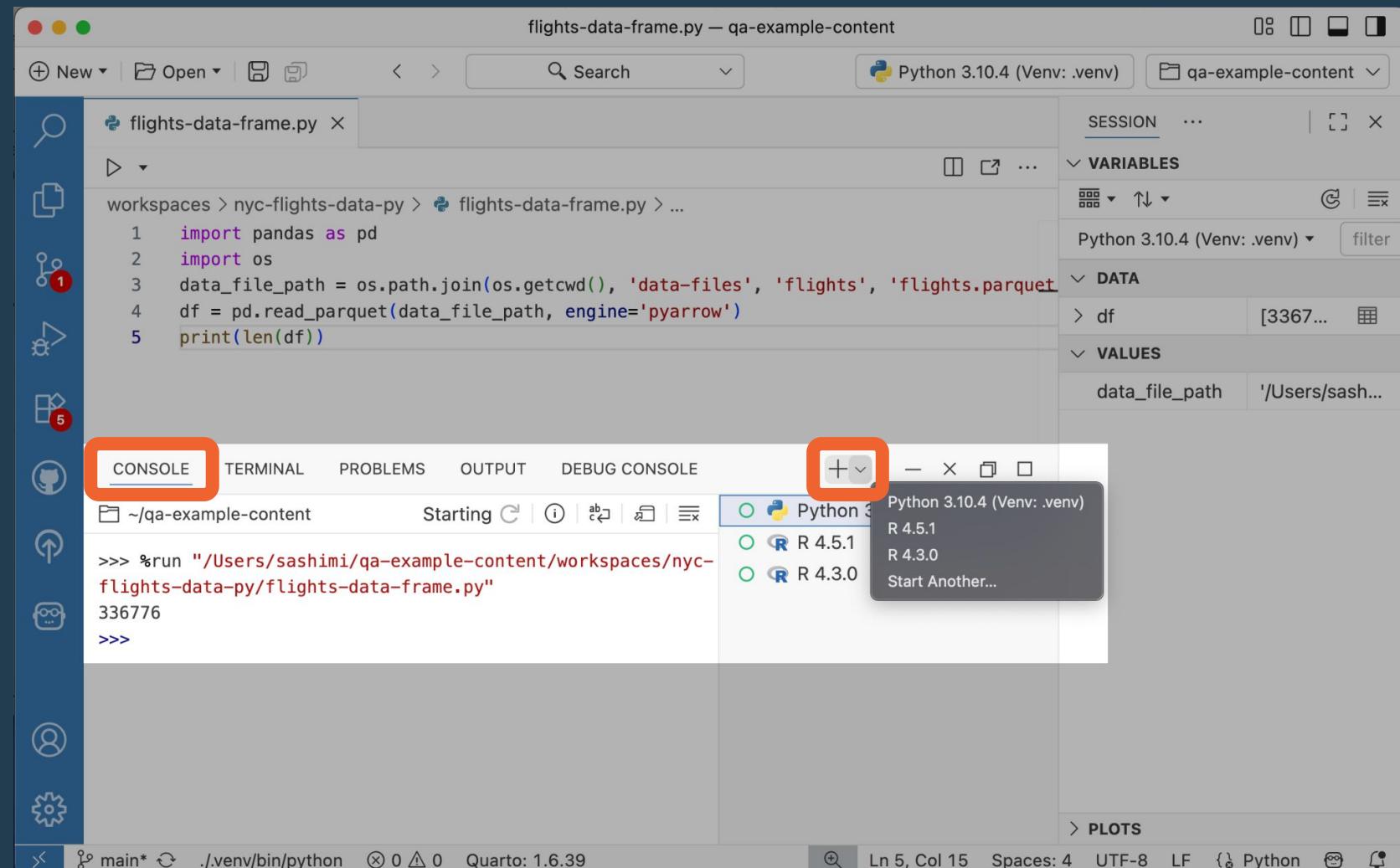
- Title Bar:** flights-data-frame.py — qa-example-content
- Toolbar:** Includes New, Open, Save, and Search buttons.
- Code Editor:** A file named "flights-data-frame.py" containing Python code:

```
1 import pandas as pd
2 import os
3 data_file_path = os.path.join(os.getcwd(), 'data-files', 'flights', 'flights.parquet')
4 df = pd.read_parquet(data_file_path, engine='pyarrow')
5 print(len(df))
```
- Variables View:** A sidebar titled "SESSION" which is highlighted with an orange box. It shows "Python 3.10.4 (Venv: .venv)".
- Data View:** Shows a table for "df": [336776 rows x 19 columns].
- Values View:** Shows the value for "data_file_path": "/Users/sashimi/qa-example-content/workspaces/nyc-flights-data-py/flights-data-frame.py".
- Console Tab:** Shows the output of running the script:

```
>>> %run "/Users/sashimi/qa-example-content/workspaces/nyc-flights-data-py/flights-data-frame.py"
336776
>>>
```
- Interpreter Selector:** At the bottom of the console tab, it shows "Python 3.10.4 (Venv: .venv)" and "R 4.5.1".
- Bottom Status Bar:** Shows the current file ("main*"), the path ("./venv/bin/python"), and the Quarto version ("Quarto: 1.6.39").

Console

- Multiple R and Python sessions are possible!



Console

- Right-click to rename sessions
- Feel free to use emojis ✨

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** polars_basic.py — qa-example-content
- File List:** flights-data-frame.py, polars_basic.py (highlighted), polaris_dataframe.py, polaris_basic.py
- Code Editor:** A Python code cell defining a DataFrame `df` with columns `foo`, `bar`, `ham`, and `a`.
- Variables View:** Shows variables like `df` and `date`.
- Session Management:** A dropdown menu in the bottom right corner lists sessions:
 - flights analysis
 - R
 - R 4.3.0
 - trying out polars (selected)A "Rename..." button is visible next to the R session entry.
- Bottom Status Bar:** Shows the current file (main*), Python interpreter path (.venv/bin/python), and Quarto version (1.6.39).

Console

- Click on the info icon for more details about your session

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** polars_basic.py — qa-example-content
- Toolbar:** New, Open, SESSION, VIEWER, etc.
- Code Cell:** A code cell containing Python code to import polars and create a DataFrame. The code is as follows:

```
1 import polars as pl
2
3 from datetime import date
4 df = pl.DataFrame(
5     {
6         "foo": [1, 2, 3],
7         "bar": [6.0, 7.0, 8.0],
8         "ham": [date(2020, 1, 2), date(2021, 3, 4), date(2022, 5, 6)],
9         "a": [None, 2, 3],
10        "b": [0.5, None, 2.5],
11        "c": [True, None, False],
12    }
13 )
14 >>>
```
- Sidebar:** A vertical sidebar with various icons, some with red numbers indicating notifications (e.g., 1, 5).
- Status Bar:** Shows the file main*, the command ./venv/bin/python, and the Quarto version 1.6.39.
- Bottom Status Bar:** Shows the line number (Ln 8), column number (Col 52), spaces (Spaces: 4), encoding (UTF-8), language (LF), and Python.
- Info Icon:** An info icon (a circle with an 'i') is highlighted with a red box in the status bar area.
- Pop-up Info Window:** A window titled "trying out polars" provides session details:
 - Session ID: python-50d8333b
 - State: idle
 - Path: ~/qa-example-content/.venv/bin/python
 - Source: Venv

It also includes links to "Show Console Output Channel", "Show Kernel Output Channel", and "Show LSP Output Channel".
- Session List:** A list of sessions on the right side of the status bar, with "trying out polars" selected.

Custom Keybindings

Run command Preferences: Open
Keyboard Shortcuts

>keyboard

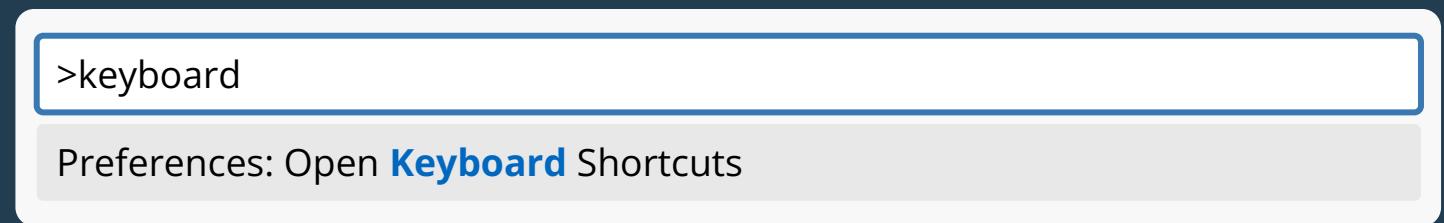
Preferences: Open **Keyboard** Shortcuts



Custom Keybindings

- Assign any Command Palette command to a hotkey

Run command Preferences: Open Keyboard Shortcuts



The screenshot shows the 'Keyboard Shortcuts' settings page. The title bar includes tabs for 'Keyboard Shortcuts X' and 'keybindings.json'. Below the title bar is a search bar with the placeholder 'Type to search in keybindings'. The main area is a table listing keybindings:

| Command | Keybinding | When | Source |
|-------------------------------|------------|--|---------------------|
| Positron Data Explorer: Copy | ⌘ C | positronDataExplorerFocused && activeEd... | System |
| Positron: Go to Next Cell | ⌘ PageDown | editorTextFocus && positron.hasCodeCell... | Positron Code Cells |
| Positron: Go to Previous Cell | ⌘ PageUp | editorTextFocus && positron.hasCodeCell... | Positron Code Cells |
| Positron: Insert Code Cell | ⇧ ⌘ I | editorTextFocus && positron.supportsCod... | Positron Code Cells |

Run command Preferences: Open Keyboard Shortcuts (JSON)

Custom Keybindings

- Add more complex keyboard shortcuts to the `keybindings.json` file

`>keyboard json`

Preferences: Open **Keyboard** Shortcuts (**JSON**)

Run command Preferences: Open Keyboard Shortcuts (JSON)

Custom Keybindings

- Use built-in `runCommands` to chain actions together (e.g. restart the terminal)

>keyboard.json

Preferences: Open **Keyboard** Shortcuts (**JSON**)

keybindings.json

```
{  
  "key": "shift+cmd+9",  
  "command": "runCommands",  
  "args": {  
    "commands": [  
      "workbench.action.terminal.kill",  
      "workbench.action.terminal.new"  
    ]  
  },  
  "when": "terminal.active"  
}
```



Making data exploration more pleasant

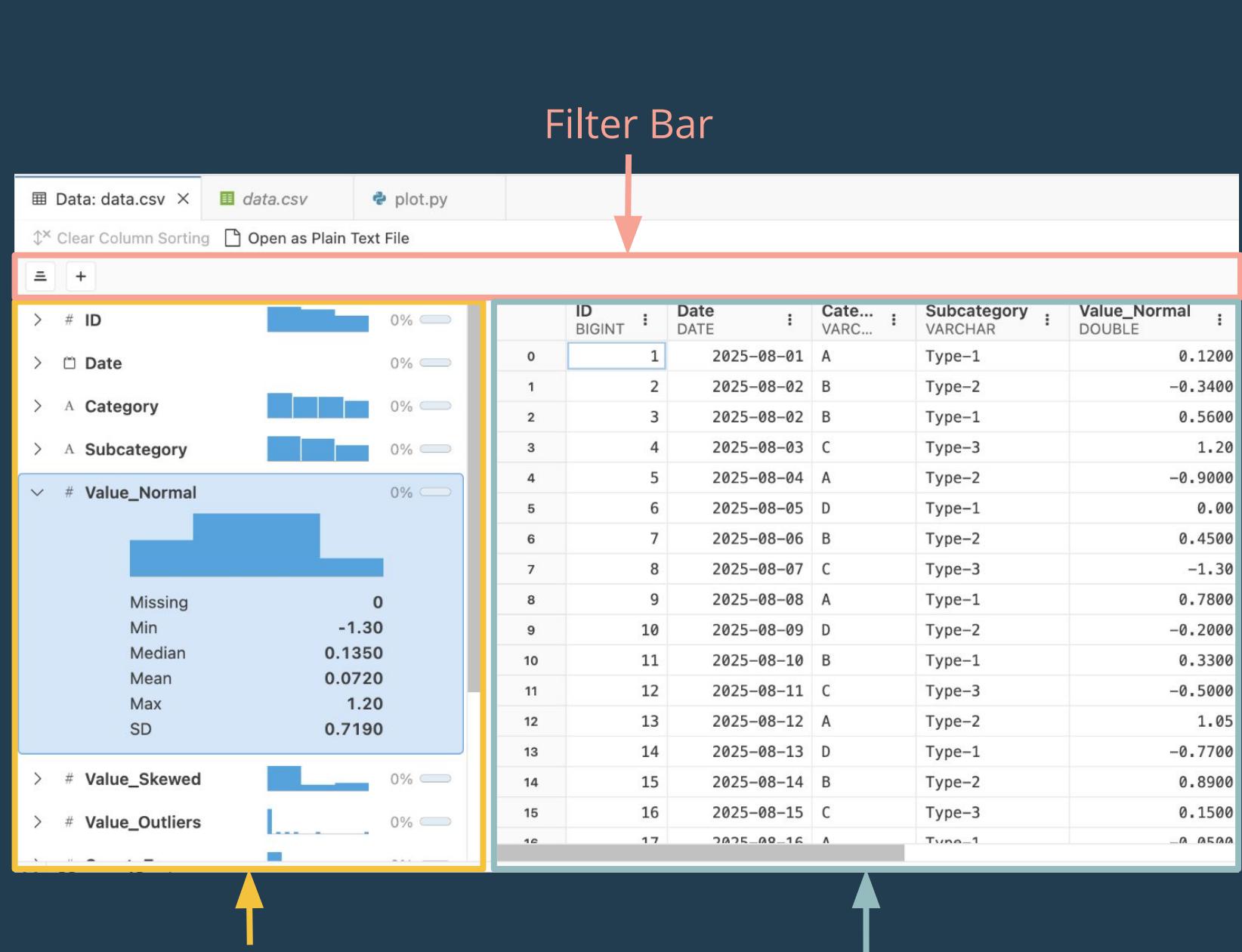
Editing and Authoring with more fluidity

posit
conf (2025)



Data Explorer

- View your data in a friendly interface



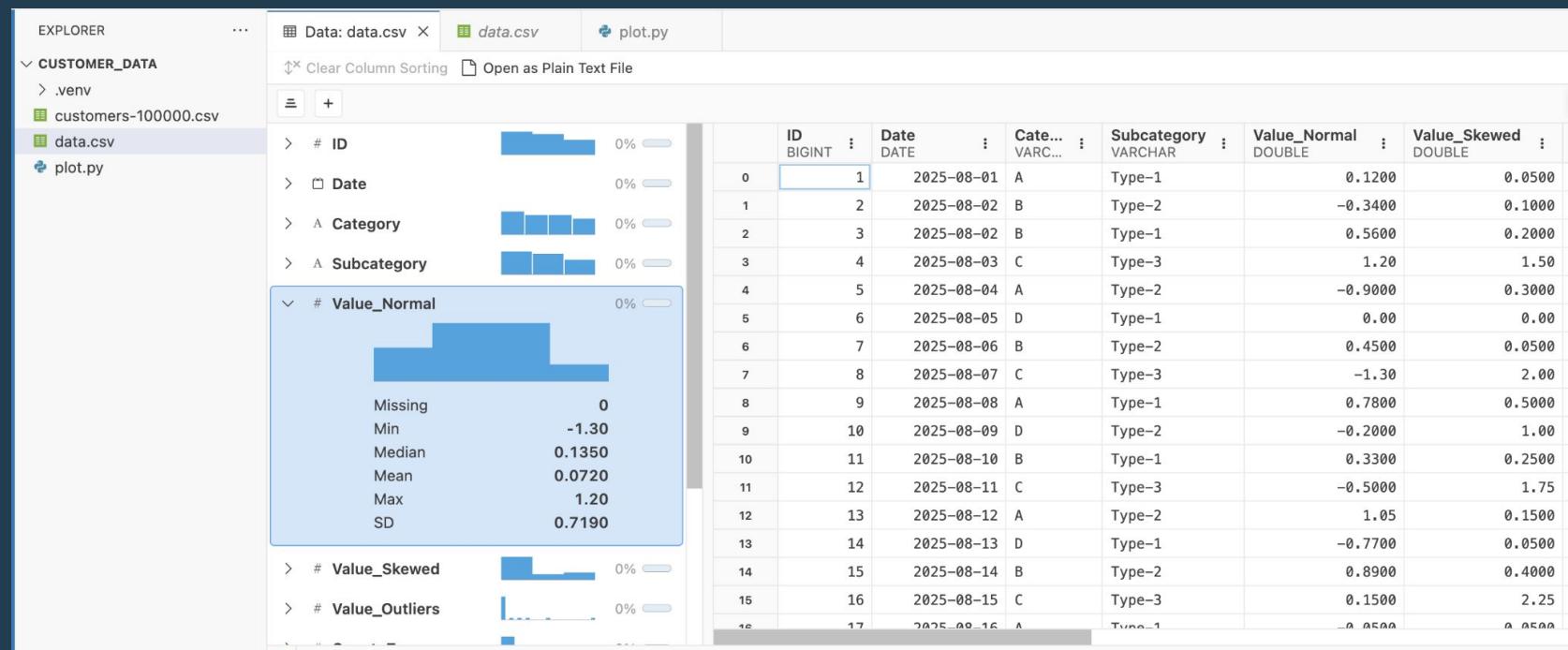
Summary Panel

Data Grid

Filter Bar

Data Explorer

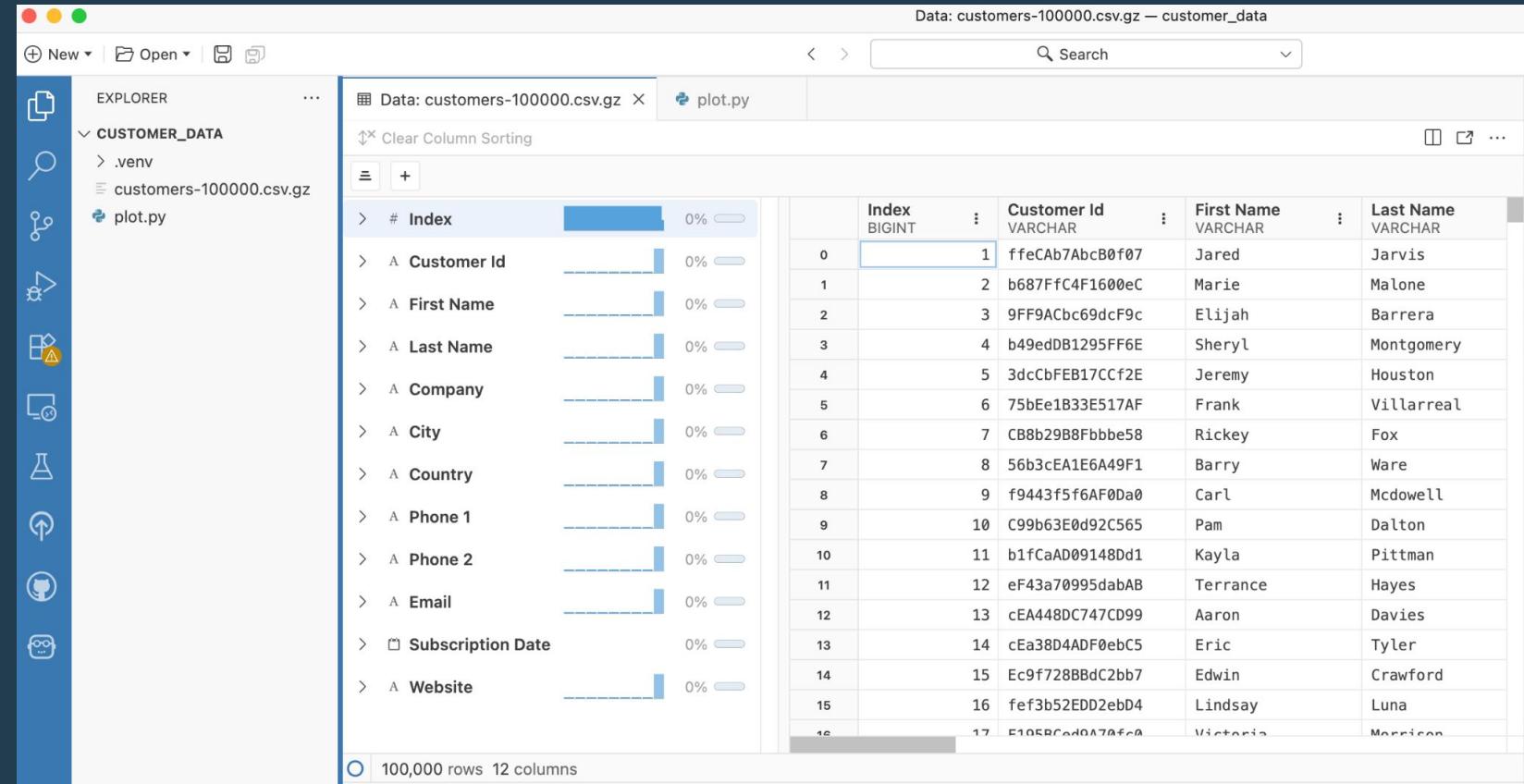
- Click to view your data files in the Data Explorer



data.csv open in the Data Explorer

Data Explorer

- Compressed data files open just as smoothly



The screenshot shows the Data Explorer interface with a dark theme. On the left is a sidebar with icons for New, Open, Explorer, CUSTOMER_DATA (which is expanded), .venv, customers-100000.csv.gz, and plot.py. The main area has a title bar "Data: customers-100000.csv.gz — customer_data". Below it is a search bar with a magnifying glass icon and a "Clear Column Sorting" button. The data view shows a table with 100,000 rows and 12 columns. The columns are: Index (BIGINT), Customer Id (VARCHAR), First Name (VARCHAR), and Last Name (VARCHAR). The first few rows of data are:

| Index | Customer Id | First Name | Last Name |
|-------|-----------------|------------|------------|
| 0 | ffeCab7AbcB0f07 | Jared | Jarvis |
| 1 | b687Ffc4F1600eC | Marie | Malone |
| 2 | 9FF9ACbc69dcF9c | Elijah | Barrera |
| 3 | b49edDB1295FF6E | Sheryl | Montgomery |
| 4 | 3dcCbFEB17CCf2E | Jeremy | Houston |
| 5 | 75bEe1833E517AF | Frank | Villarreal |
| 6 | CB8b29B8Fbbbe58 | Rickey | Fox |
| 7 | 56b3cEA1E6A49F1 | Barry | Ware |
| 8 | f9443f5f6AF0Da0 | Carl | Mcdowell |
| 9 | C99b63E0d92C565 | Pam | Dalton |
| 10 | b1fCaAD09148Dd1 | Kayla | Pittman |
| 11 | eF43a70995dabAB | Terrance | Hayes |
| 12 | cEA448DC747CD99 | Aaron | Davies |
| 13 | cEA38D4ADF0ebC5 | Eric | Tyler |
| 14 | Ec9f728BBdC2bb7 | Edwin | Crawford |
| 15 | fef3b52EDD2ebD4 | Lindsay | Luna |
| 16 | E105RCod0A7afca | Victoria | Morrison |
| 17 | | | |

At the bottom left of the data view, there is a circular icon with a dot and the text "100,000 rows 12 columns".

customers-100000.csv.gz open in the Data Explorer

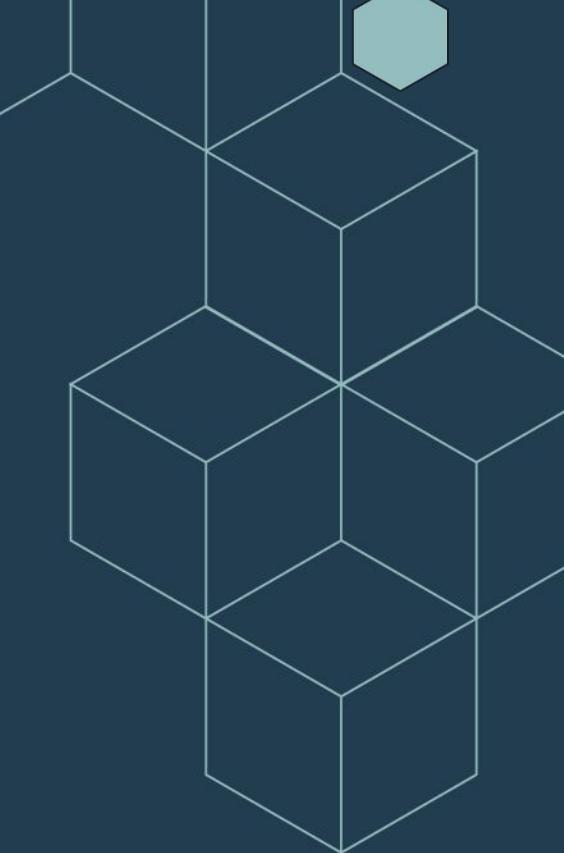
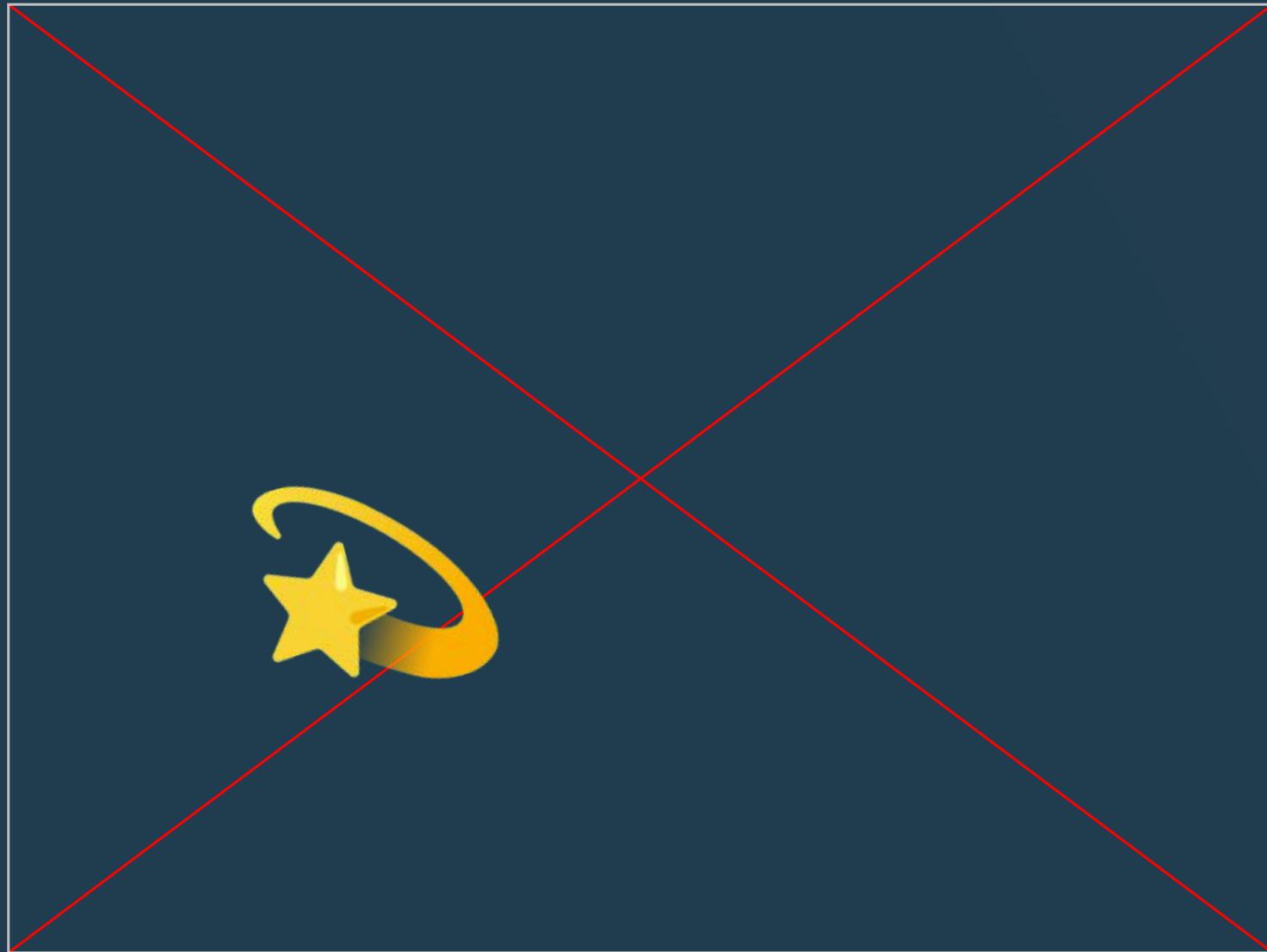
Data Explorer

- Open plain text as well

The screenshot shows a Data Explorer interface with a sidebar labeled 'CUSTOMER_DATA' containing files: '.venv', 'customers-100000.csv', 'data.csv', and 'plot.py'. The 'data.csv' file is selected. In the main area, there are two tabs: 'Data: data.csv' and 'data.csv'. The 'data.csv' tab is active, showing the contents of the CSV file as plain text. A red box highlights the 'Open as Plain Text File' button in the top right corner of the main window.

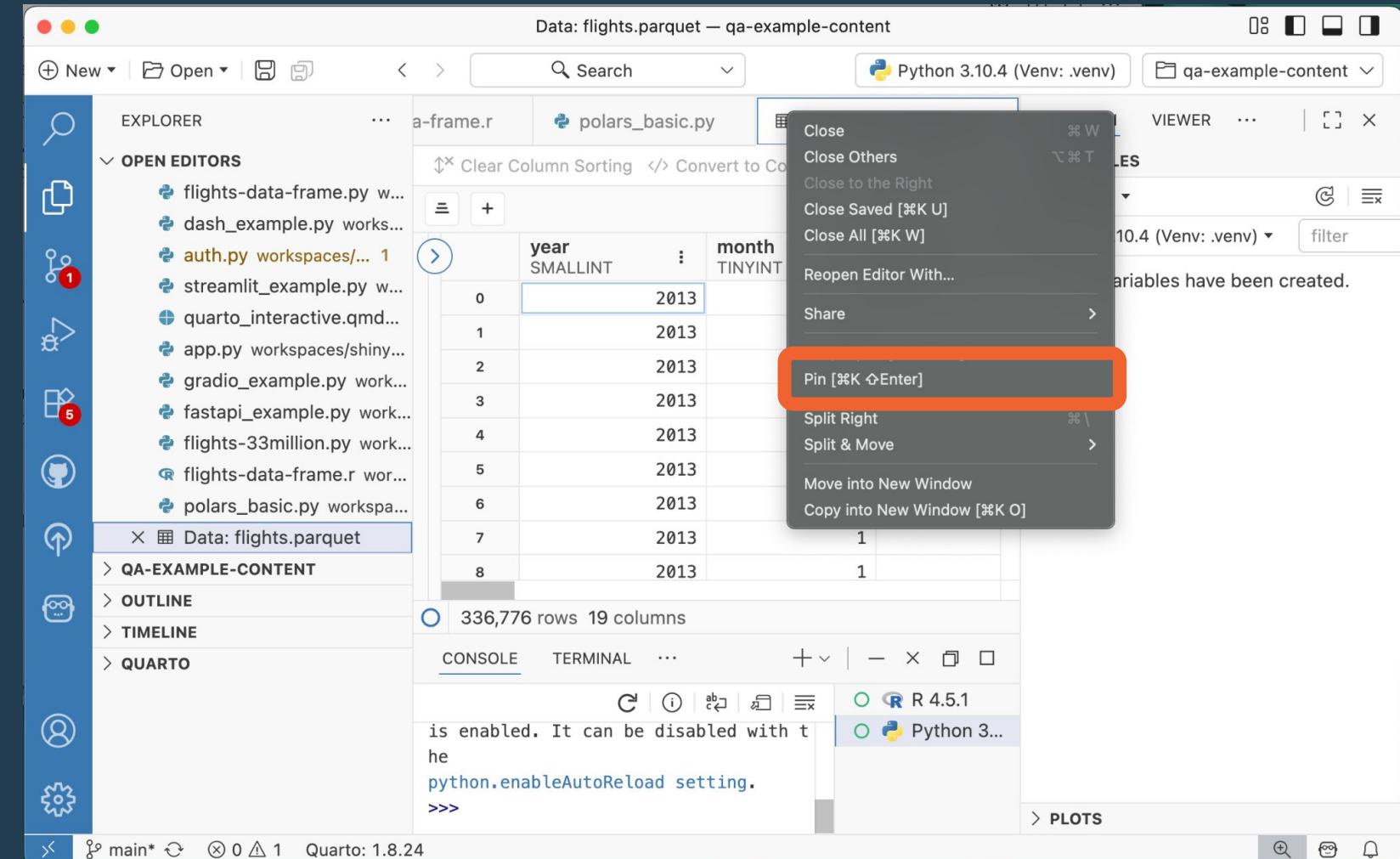
| ID | Date | Category | Subcategory | Value_Normal | Value_Skewed | Value_Outliers | Count_Zeros | Flag | Text_Memo | Latitude | Longitude | Missing_Ratio |
|----|------------|----------|-------------|--------------|--------------|----------------|-------------|-------|---------------------|----------|-----------|---------------|
| 1 | 2025-08-01 | A | Type-1 | 0.12 | 0.05 | 20.0 | 0 | TRUE | "First note" | 42.80 | -70.87 | 0.5 |
| 2 | 2025-08-02 | B | Type-2 | -0.34 | 0.10 | 0.30 | 5 | FALSE | ,42.81,-70.86, | | | |
| 3 | 2025-08-02 | B | Type-1 | 0.56 | 0.20 | 0.10 | 0 | TRUE | "Quick check" | 42.82 | -70.85 | 0.1 |
| 4 | 2025-08-03 | C | Type-3 | 1.20 | 1.50 | 100.00 | 0 | FALSE | "An outlier maybe" | 42.83 | -70.84 | 0.0 |
| 5 | 2025-08-04 | A | Type-2 | -0.90 | 0.30 | 0.20 | 3 | TRUE | "",42.84,-70.83 | 0.2 | | |
| 6 | 2025-08-05 | D | Type-1 | 0.00 | 0.00 | 0.00 | 0 | FALSE | "Zeroes everywhere" | 42.85 | -70.82 | 0.9 |
| 7 | 2025-08-06 | B | Type-2 | 0.45 | 0.05 | 5.00 | 0 | TRUE | "Mid-range" | 42.86 | -70.81 | |
| 8 | 2025-08-07 | C | Type-3 | -1.30 | 2.00 | 500.00 | 10 | FALSE | "Extreme outlier" | 42.87 | -70.80 | 0.7 |
| 9 | 2025-08-08 | A | Type-1 | 0.78 | 0.50 | 0.20 | 2 | TRUE | "Near mean" | 42.88 | -70.79 | 0.3 |
| 10 | 2025-08-09 | D | Type-2 | -0.20 | 1.00 | 0.00 | 0 | FALSE | ,42.89,-70.78 | 0.4 | | |
| 11 | 2025-08-10 | B | Type-1 | 0.33 | 0.25 | 1.00 | 1 | TRUE | "Evening" | 42.90 | -70.77 | 0.6 |
| 12 | 2025-08-11 | C | Type-3 | -0.50 | 1.75 | 200.00 | 0 | FALSE | "Mid outlier" | 42.91 | -70.76 | 0.0 |
| 13 | 2025-08-12 | A | Type-2 | 1.05 | 0.15 | 0.50 | 4 | TRUE | "Note 13" | 42.92 | -70.75 | 0.8 |
| 14 | 2025-08-12 | D | Type-1 | 0.77 | 0.25 | 0.00 | 0 | FALSE | ,42.93,-70.74 | 0.3 | | |

Has this ever happened to you?



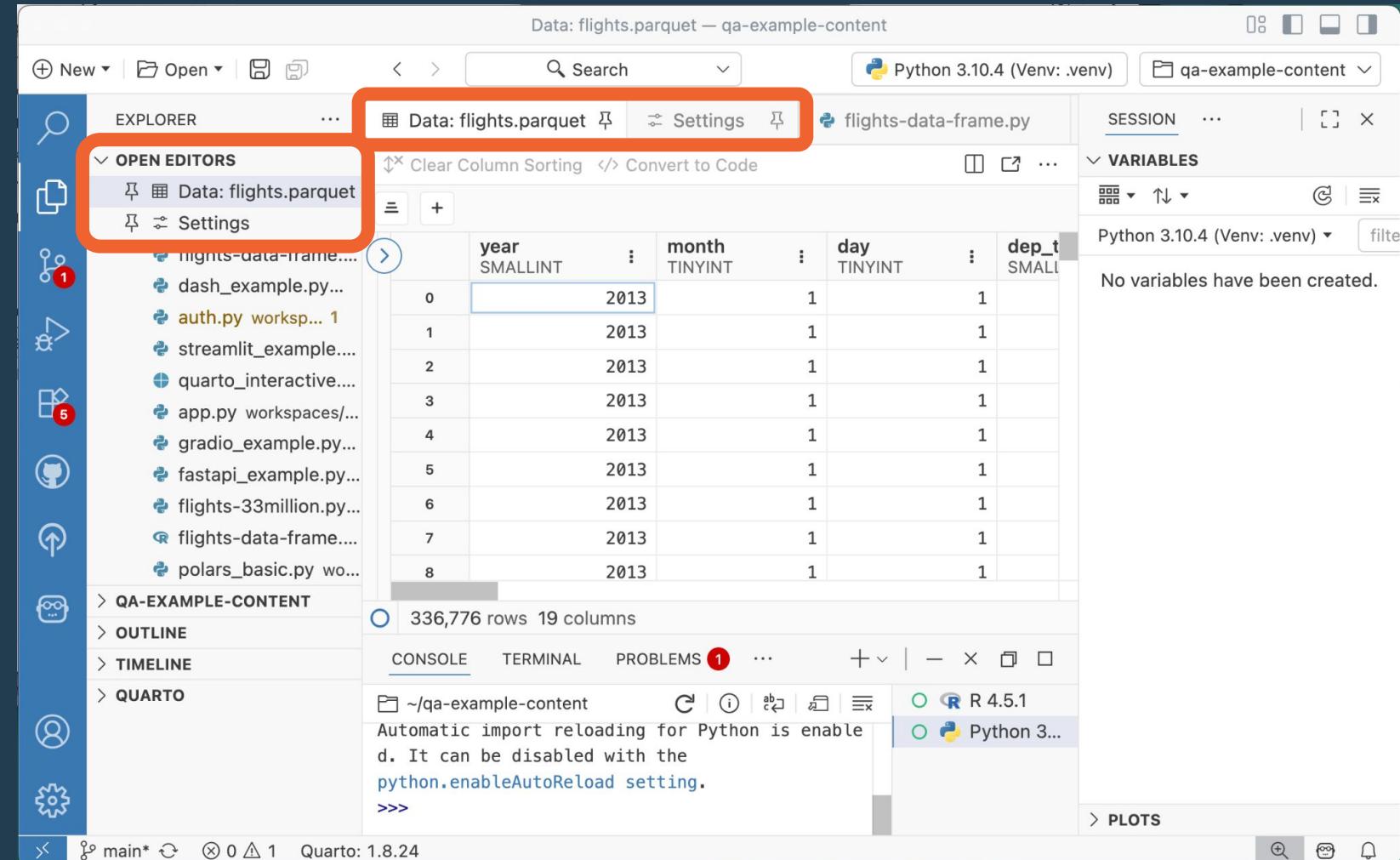
Pinned editor tabs

- Right-click on an editor tab and select Pin



Pinned editor tabs

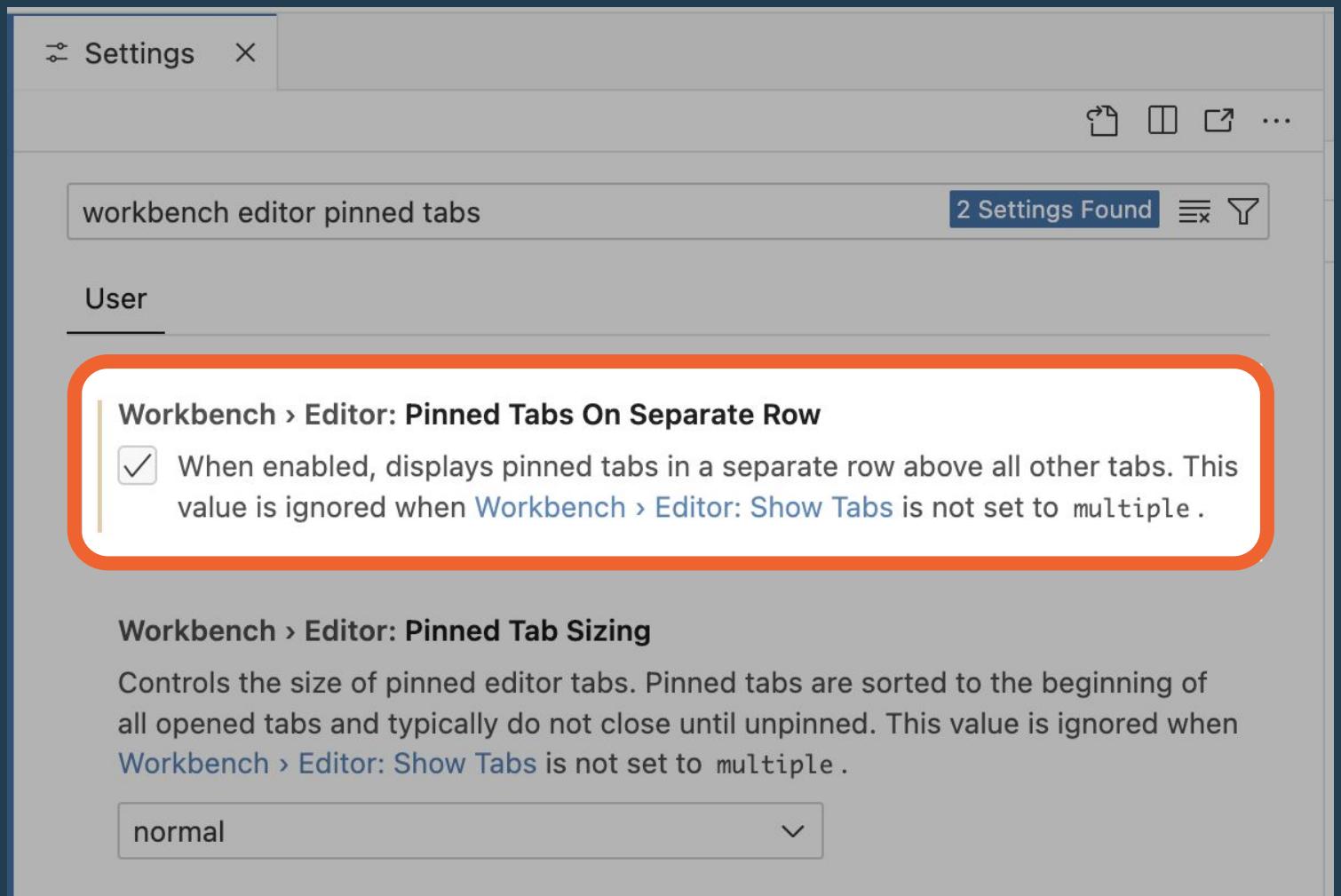
- Right-click on an editor tab and select Pin



Pinned editor tabs

- Keep pinned tabs on a separate row

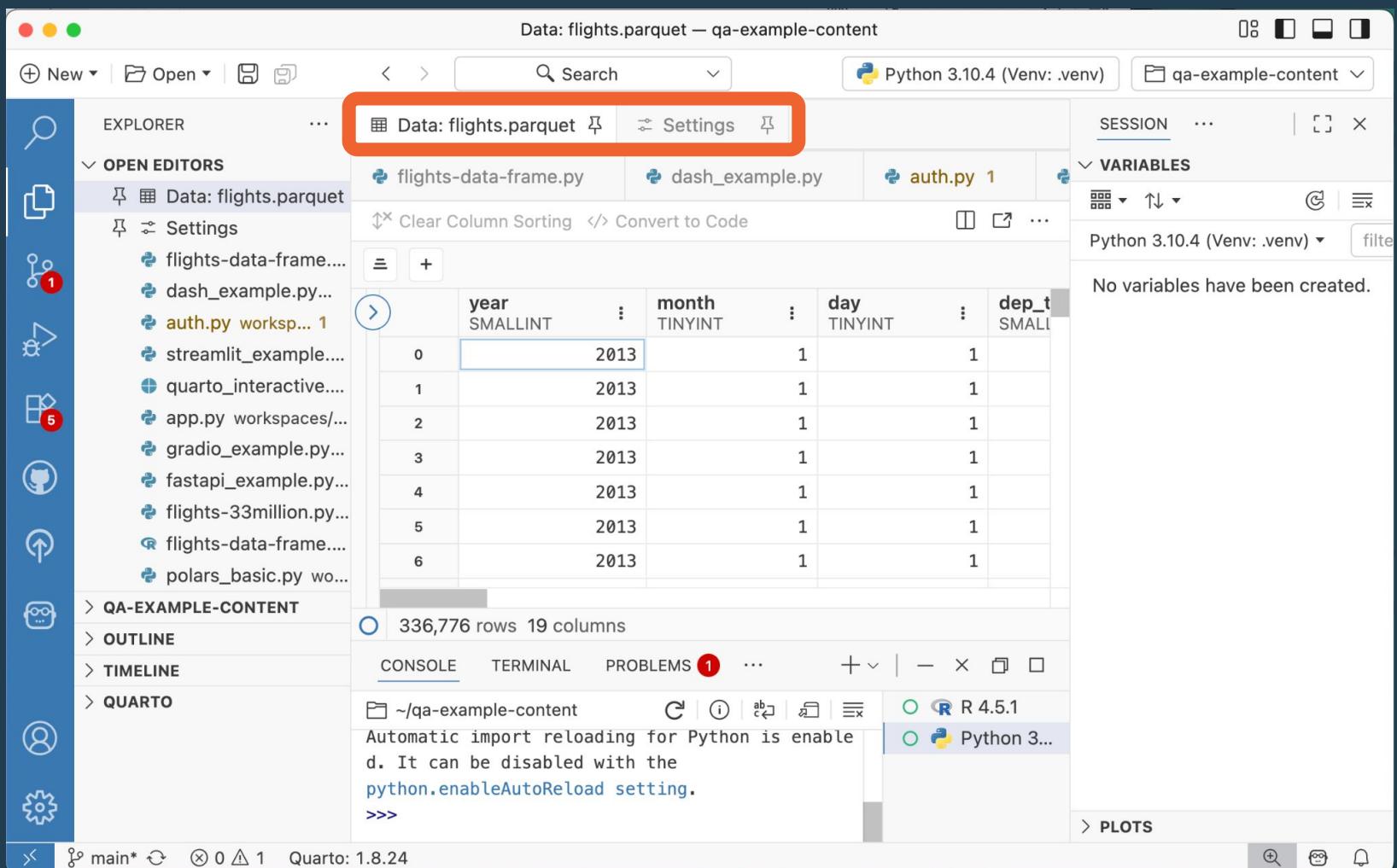
In Settings, search for and enable Workbench > Editor: Pinned Tabs on Separate Row



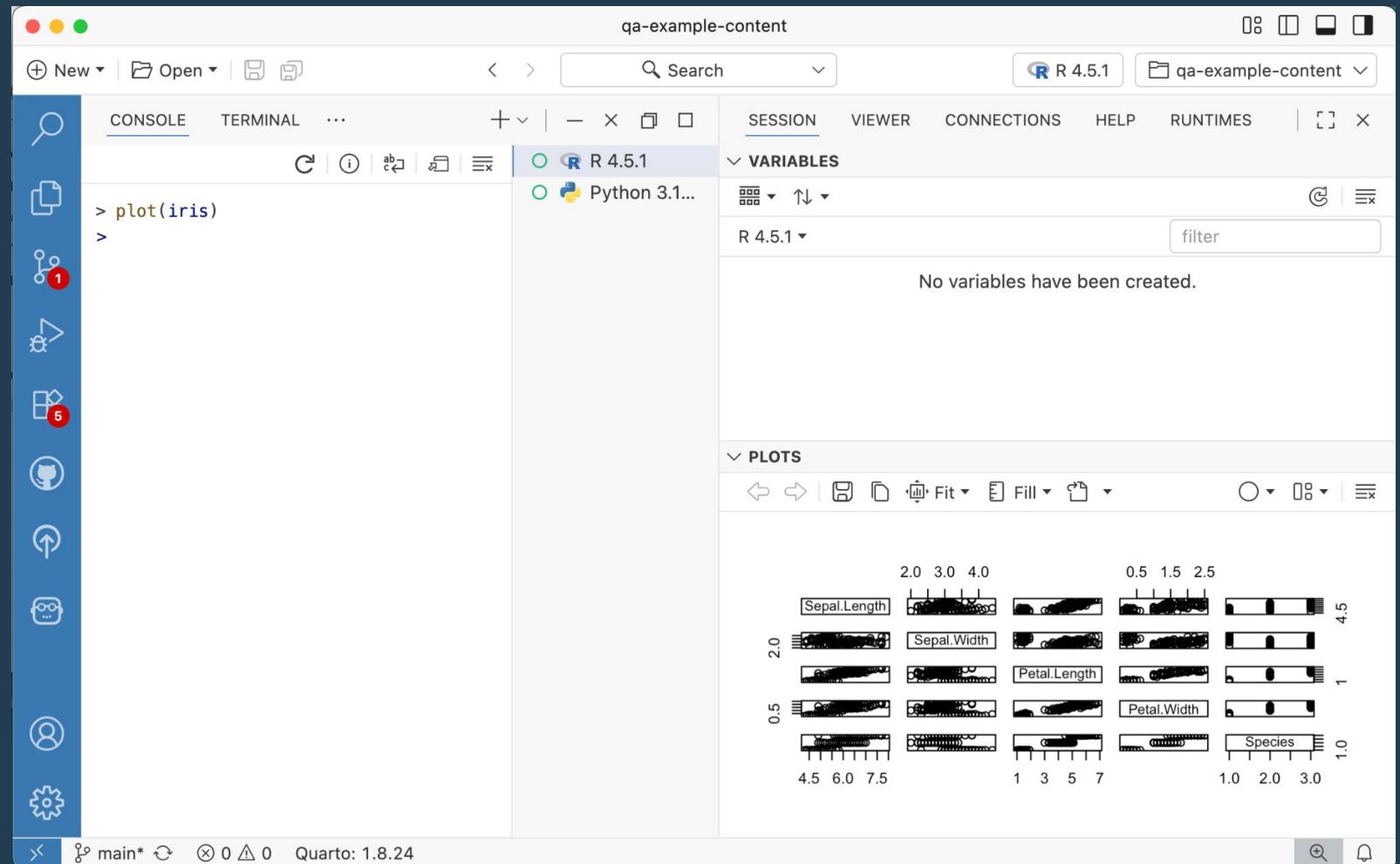
Pinned editor tabs

- Keep pinned tabs on a separate row

In Settings, search for and enable Workbench > Editor: Pinned Tabs on Separate Row

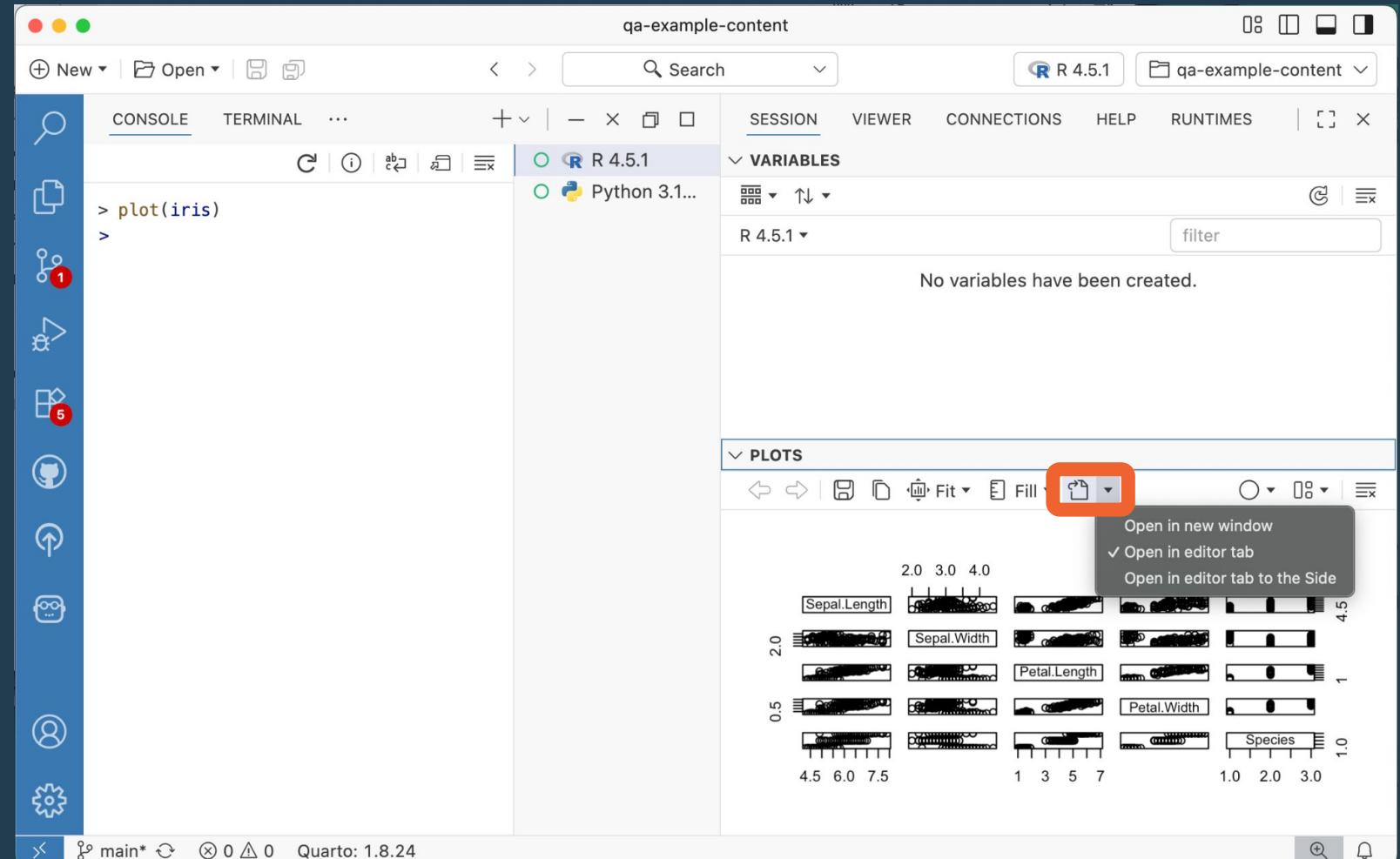


Plots



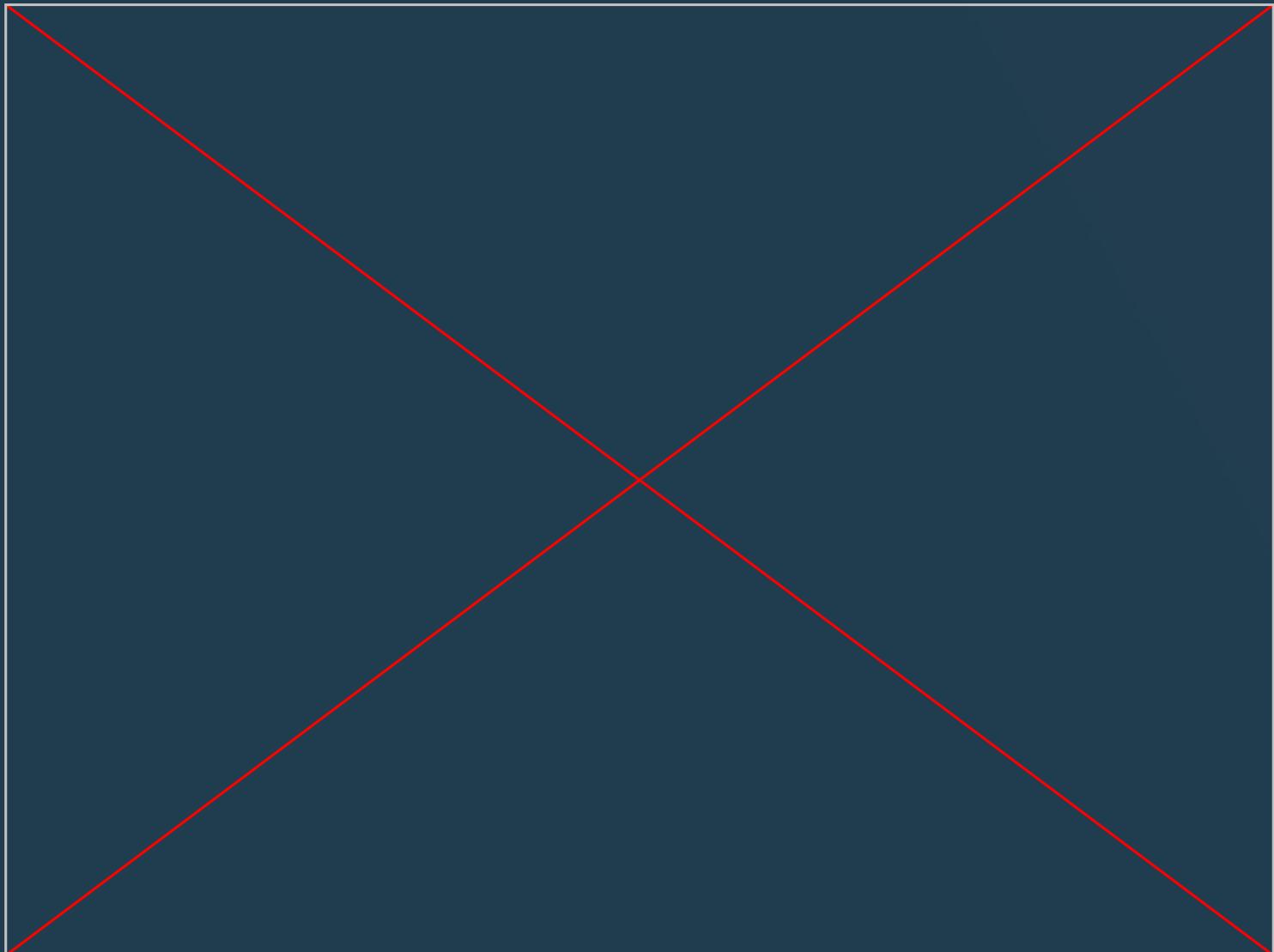
Plots

- Open plots in a new window or an editor tab for a larger viewing area





spontaneous theme change!



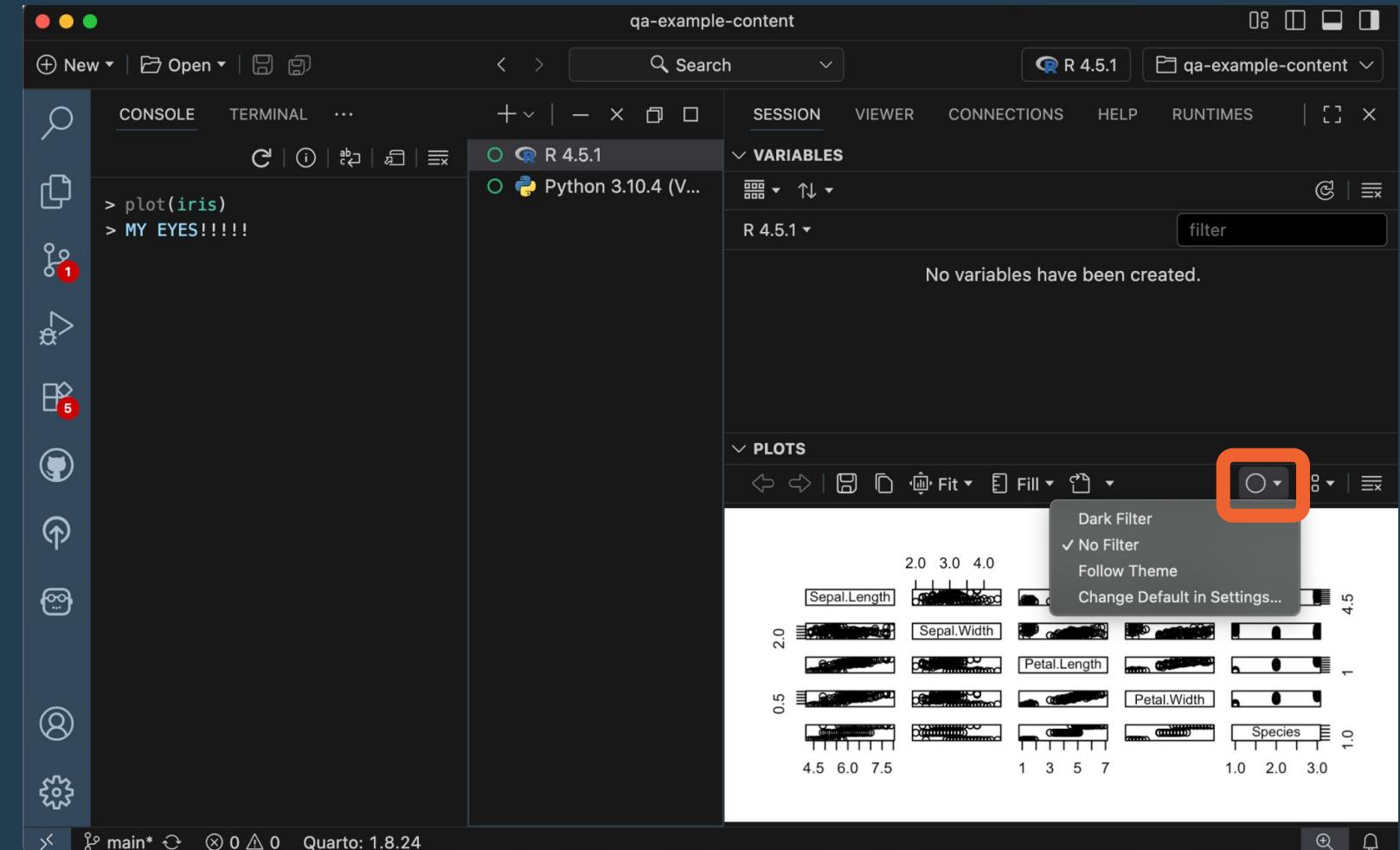


spontaneous theme change!

A screenshot of the Quarto 1.8.24 interface. The interface has a dark mode theme. On the left is a sidebar with various icons: a magnifying glass, a document, a circular arrow, a grid with a red dot, a bar chart, a pie chart, a scatter plot, a person, and a gear. The main area shows a Quarto project titled "qa-example-content". The "CONSOLE" tab is active, displaying the command `> plot(iris)` followed by the text "MY EYES!!!!". The "SESSION" tab shows two sessions: "R 4.5.1" and "Python 3.10.4 (V...)" (partially visible). The "VARIABLES" section shows a dropdown menu set to "R 4.5.1" with the message "No variables have been created.". The "PLOTS" section displays a grid of 15 box plots for the "iris" dataset, with labels for Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. A context menu is open over one of the plots, listing options: "Dark Filter" (unchecked), "No Filter" (checked), "Follow Theme", and "Change Default in Settings...". The bottom status bar shows "main*" and "Quarto: 1.8.24".

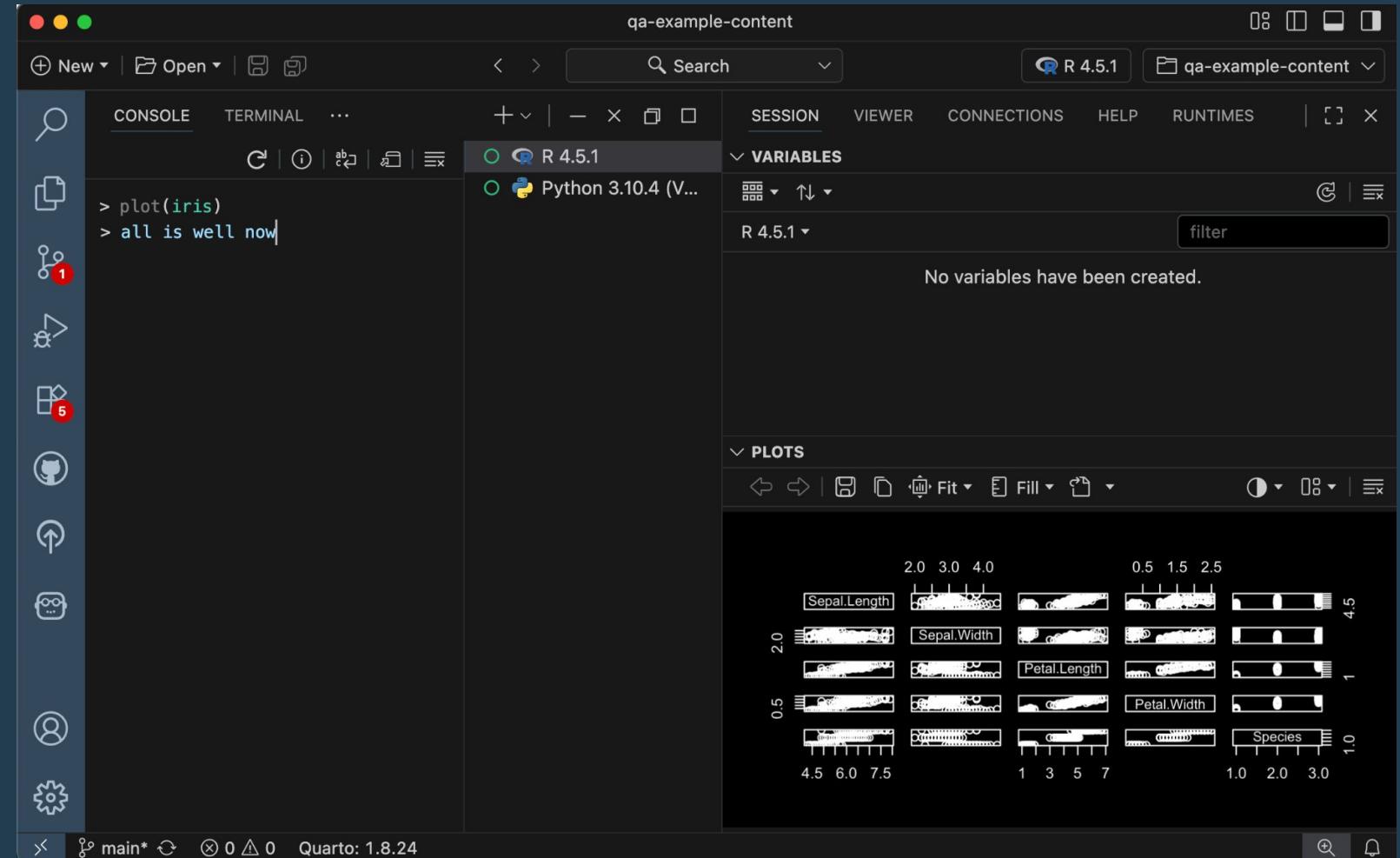
Plots

- Set the dark filter setting to Dark Filter for dark plots always, or set the filter to Follow Theme



Plots

- Set the dark filter setting to Dark Filter for dark plots always, or set the filter to Follow Theme



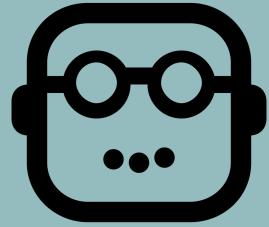


Thoughtful Pairing with Positron Assistant

Enhancing your analysis with AI

posit
conf (2025)





What is Positron Assistant?

An AI client providing LLM integration in Positron

- Chat and inline completions
- Great for help with debugging, refactoring, and getting suggestions for next steps



Context beyond files on disk

- Aware of active console/runtime
- Aware of structure of in-memory objects
- Ask questions about your variables and console session

The screenshot shows the positron IDE interface. On the left, there's a sidebar with various icons for file operations, a search bar, and a 'CHAT' section titled 'Anthropic'. The main area has tabs for 'seattle-temperatures.py' and 'matplotlib-lib-ex.py'. The 'matplotlib-lib-ex.py' tab is active, displaying the following code:

```
5 population_by_continent = {
6     "africa": [1044, 1273],
7     "americas": [340, 425, 519, 619, 727, 840,
8         943, 1066],
9     "asia": [1394, 1686, 2120, 2625, 3202, 3714,
10        4169, 4560],
11    "europe": [220, 253, 276, 295, 310, 303,
12        294, 293],
13    "oceania": [12, 15, 19, 22, 26, 31, 36, 39],
14 }
15
16 fig, ax = plt.subplots()
17 ax.stackplot(
18     year,
19     population_by_continent.values(),
20     labels=population_by_continent.keys(),
21     alpha=0.8,
```

The 'CONSOLE' tab shows the following command and its error:

```
... # Create legend with transparent background
... legend = ax.legend(loc="upper left", framealpha=0)
... ax.set_title("World population")
... ax.set_xlabel("Year")
... ax.set_ylabel("Number of people (millions)")
...
... plt.show()
>>> plt.SHOW()
```

An 'AttributeError' is shown in red:

```
AttributeError
: module 'matplotlib.pyplot' has no attribute 'SHOW'
```

The 'SESSION' tab shows the Python environment details:

```
Python 3.12.0 (Venv: .venv)
```

The 'VALUES' tab lists variables and their types:

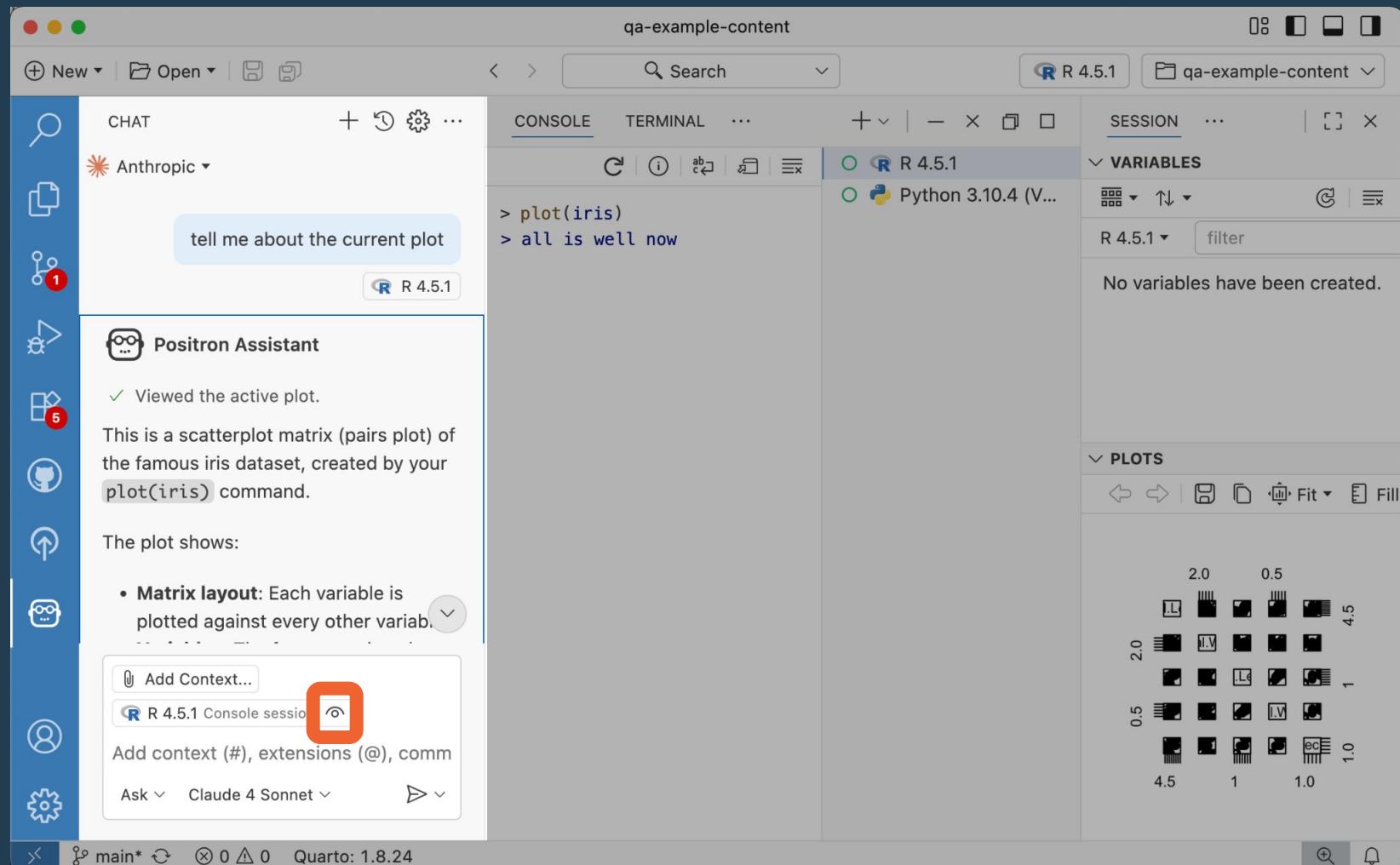
- > ax <Axes: title={'center': 'World population', 'size': 10, 'color': 'black', 'fontStyle': 'normal', 'fontWeight': 'bold', 'baseline': 'bottom', 'align': 'center', 'label': 'Year'}>
- > fig <Figure size 1163.52x786.24 with 1 Axes>
- > legend <matplotlib.legend.Legend object at 0x7f3d1c0000>
- > population_by_cc <'africa': [228, 284, 365, 441, 527, 619, 727, 840, 943, 1066], 'america': [340, 425, 519, 619, 727, 840, 943, 1066], 'asia': [1394, 1686, 2120, 2625, 3202, 3714, 4169, 4560], 'europe': [220, 253, 276, 295, 310, 303, 294, 293], 'oceania': [12, 15, 19, 22, 26, 31, 36, 39]>
- > year [1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020]

The 'PLOTS' tab displays a stacked area chart titled 'World population' showing the number of people in millions from 1950 to 2020, categorized by continent.

Configuring access to session context

- Click the eye icon to enable or disable session context

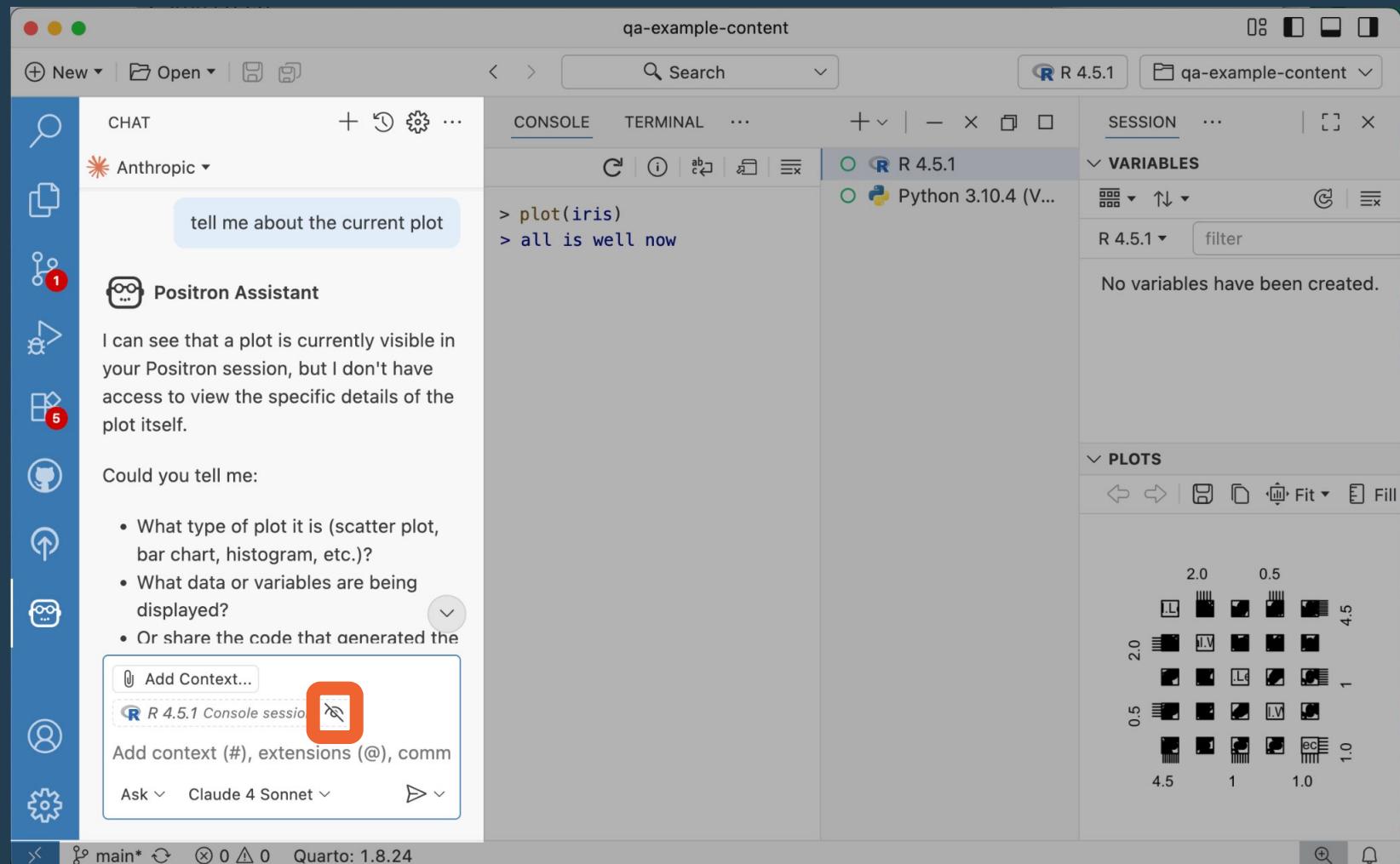
Asking about the current plot with Console session **enabled** in context



Configuring access to session context

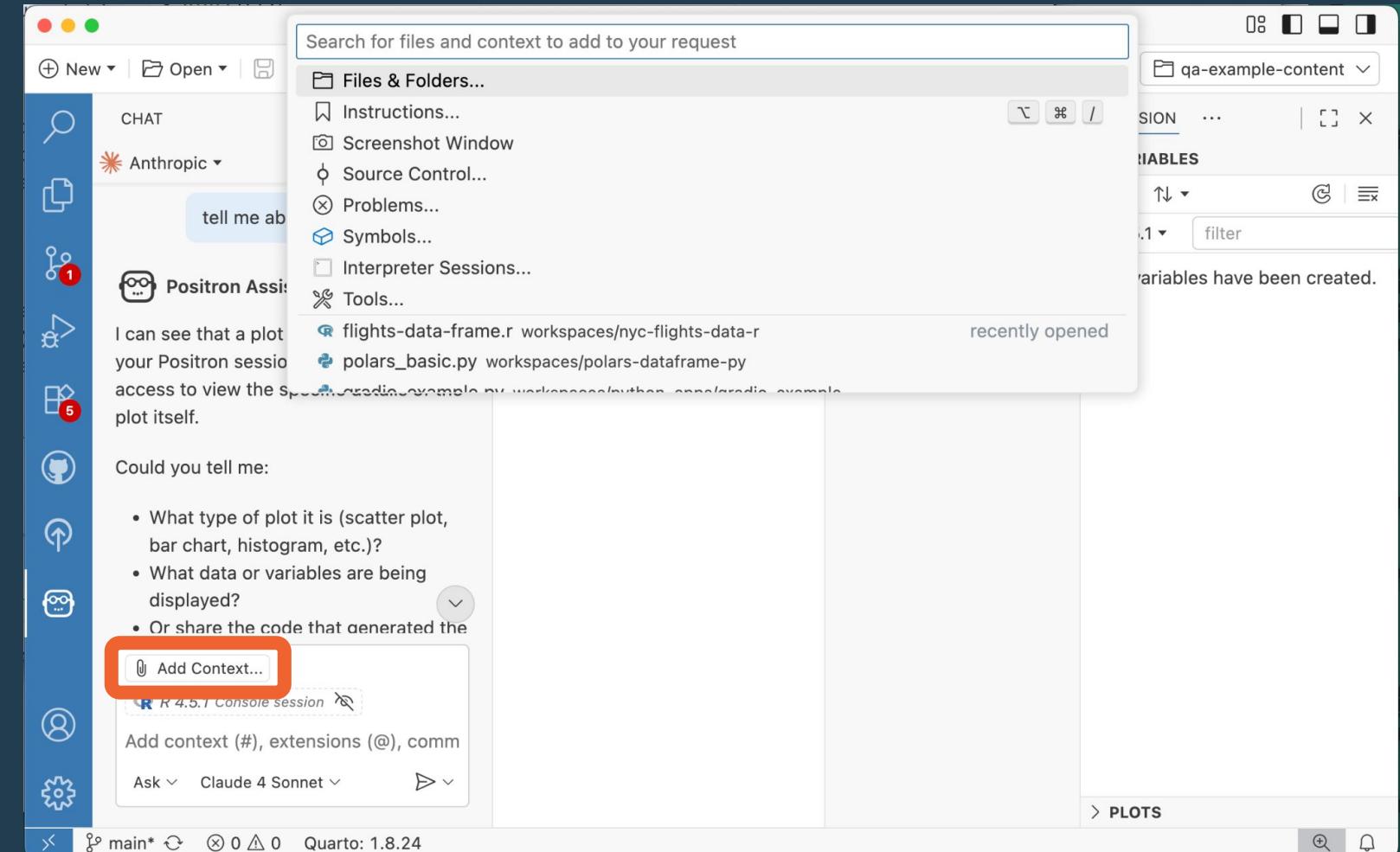
- Click the eye icon to enable or disable session context

Asking about the current plot with Console session **disabled** in context



Adding more context

- Click the Add Context... button in the chat input box to attach context to your request



Tools

- Retrieve helpful context to include with your prompt

The current plot is a stacked area chart showing the world population by continent from 1950 to 2018. Here's what's good about it:

1. It effectively shows the total world population growth over time
2. The contribution of each continent is clearly visible
3. The legend is well-placed and transparent, not blocking the data
4. The axes are properly labeled with clear units

Here are some suggested improvements:

1. Add grid lines to make it easier to read specific values:

Tools

- Look for a green checkmark, which denotes that a tool call has been made

The screenshot shows a Jupyter Notebook environment with several panels:

- Left Panel (CHAT):** An AI interface from Anthropic named Claude 3.5 Sonnet v2 is providing feedback on a plot analysis.
- Middle Panel (Code Cell):** A Python script titled "temperatures.py" is displayed. It contains code to generate a stacked area chart of world population by continent over time.
- Right Panel (Variables):** A list of variables and their values is shown, including "population_by_continent" and "year".
- PLOTS Panel:** A stacked area chart titled "World population" is displayed, showing the number of people in millions from 1950 to 2018, categorized by continent.

A red arrow points to a tooltip in the AI panel that says "Viewed the active plot." This tooltip is highlighted with a red box. A circular callout also highlights the text "let me analyze the current improvements." in the AI response.

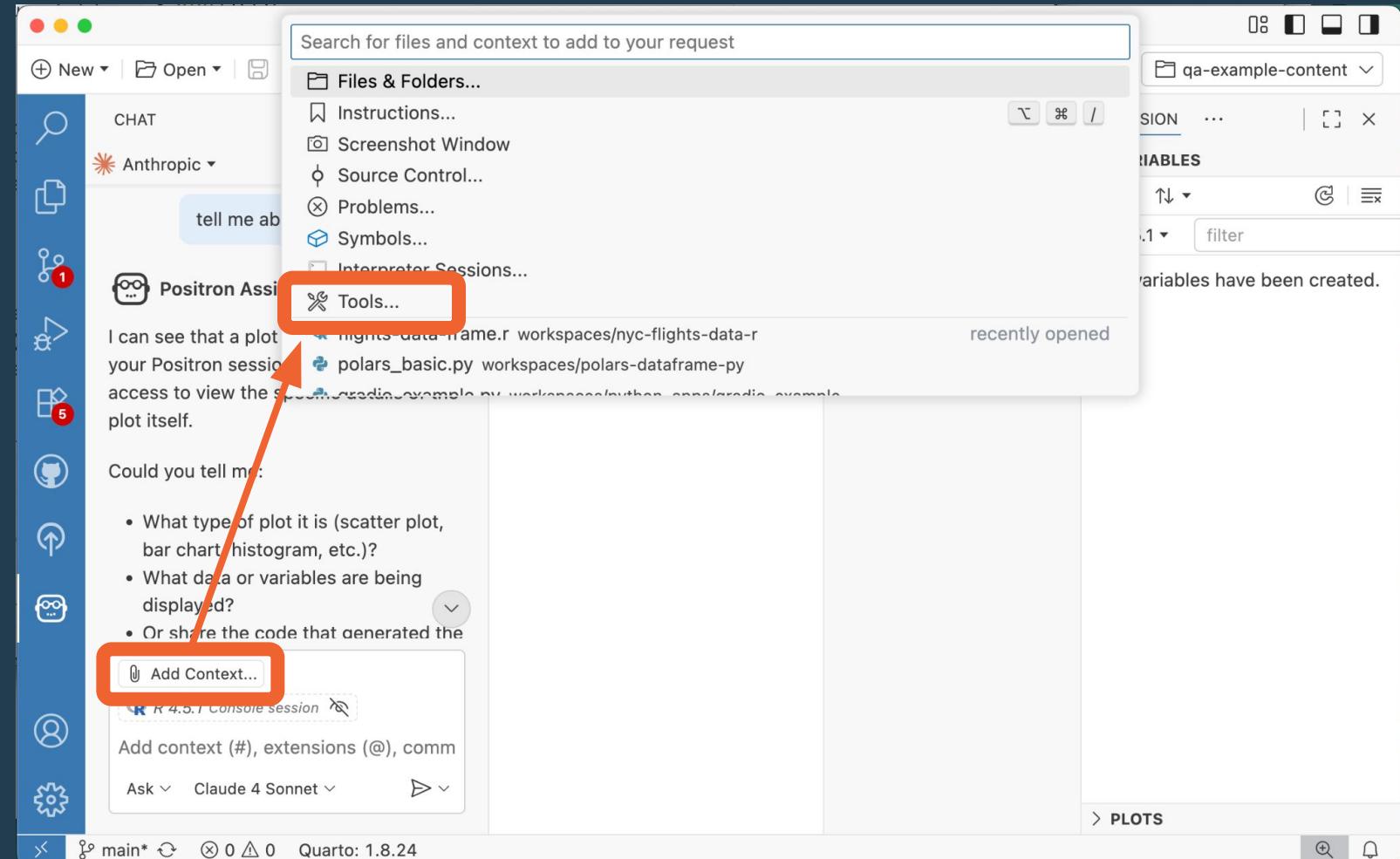
```
5 population_by_continent = [500, 477, 551, 614, 1044, 1275],  
6 ... "americas": [340, 425, 519, 619, 727, 840, 943, 1006],  
7 ... "asia": [1394, 1686, 2120, 2625, 3202, 3714, 4169, 4560],  
8 ... "europe": [220, 253, 276, 295, 310, 303, 294, 293],  
9 ... "oceania": [12, 15, 19, 22, 26, 31, 36, 39],  
10 ...  
11 }  
12  
13 fig, ax = plt.subplots()  
14 ax.stackplot(  
...  
... legend = ax.legend(loc="upper left", framealpha=0)  
... ax.set_title("World population")  
... ax.set_xlabel("Year")  
... ax.set_ylabel("Number of people (millions)")  
... plt.show()  
>>>
```

./venv/bin/python ./temperatures.py

Ln 27, Col 1 (779 selected) Spaces: 4 UTF-8 LF Python

Attaching tools as context

- Click the Add Context... button in the chat input box to attach context to your request
- Then, click the Tools... option to view the tools available as attachments



Tool: Help with R Packages and Vignettes

- Get help which directly references package documentation and vignettes

CHAT

Anthropic ▾

how else can I sort with arrange?

R-test.R:1-16 R 4.5.0

Positron Assistant used /explain (rerun without)

> Used 1 reference

I'll explain different ways to sort data using the `arrange()` function from the `dplyr` package, which you're already using in your code.

Let me first look at the help page for `arrange()` to provide you with comprehensive information:

✓ Using "Get an R Help Page"

Ways to Sort with `arrange()`

Based on the documentation and looking at your current code, here are various ways you can use `arrange()`:

1. Basic Sorting (Ascending Order)

```
# Sort by a single column in ascending order (default)
mpg %>% arrange(hwy)

# Sort by multiple columns
mpg %>% arrange(class, hwy)
```

2. Descending Order (which you're already using)

```
# Using desc() for descending order
mpg %>% arrange(desc(hwy))

# Multiple columns with mixed ordering
mpg %>% arrange(class, desc(hwy))
```

Git integration

- Ask Assistant to explain the git tracked #changes you've made to your files

CHAT + ⏪ ⚙️ ...

 Anthropic ▾

tell me about the `#changes` I've made

 changes  Python 3.10.4 (Venv: .venv)

 Positron Assistant

✓ Using "Get changed files"

You have one modified file in your workspace:

**workspaces/python_apps/gradio_example
/gradio_example.py**

The change you made was to the `image_classifier` function. You expanded the return dictionary from just two classes (`'cat': 0.3, 'dog': 0.7`) to four classes by adding:

- `'axolotl': 0.15`
- `'kangaroo': 0.25`

This modifies the mock image classifier to return predictions for four different animal categories instead of just cats and dogs.

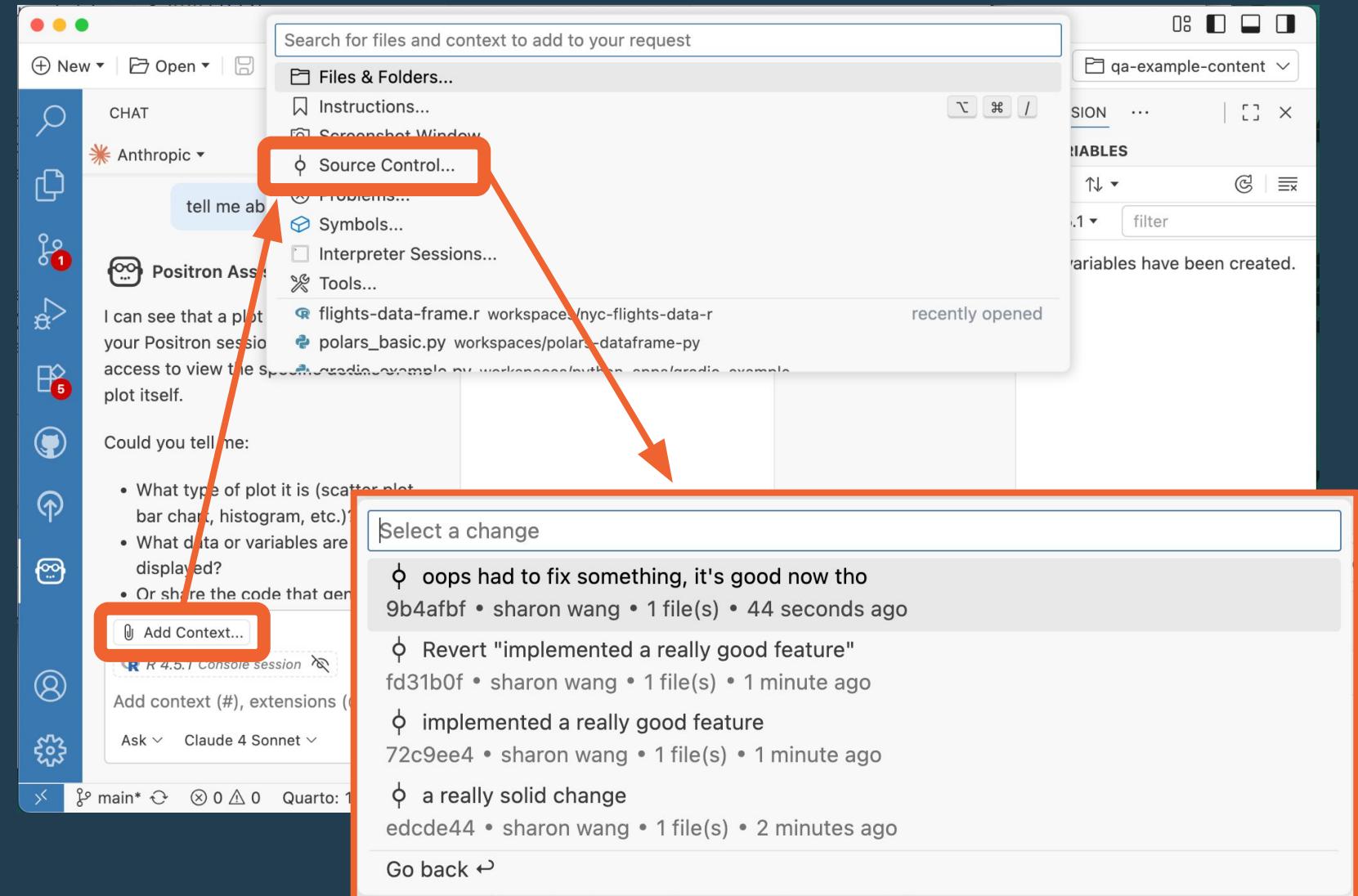
Git integration

- From the Source Control tab, click the ✨ sparkles icon to generate a commit message

```
1 1 import gradio as gr
2 2 def image_classifier(inp):
3 3     - return {'cat': 0.3, 'dog': 0.7}
4 3+     return {'cat': 0.3, 'dog': 0.7, 'axolotl': 0.15, 'kangaroo': 0.1}
5 4 demo = gr.Interface(fn=image_classifier, inputs="image", outputs=
6 5 demo.launch()
```

Git integration

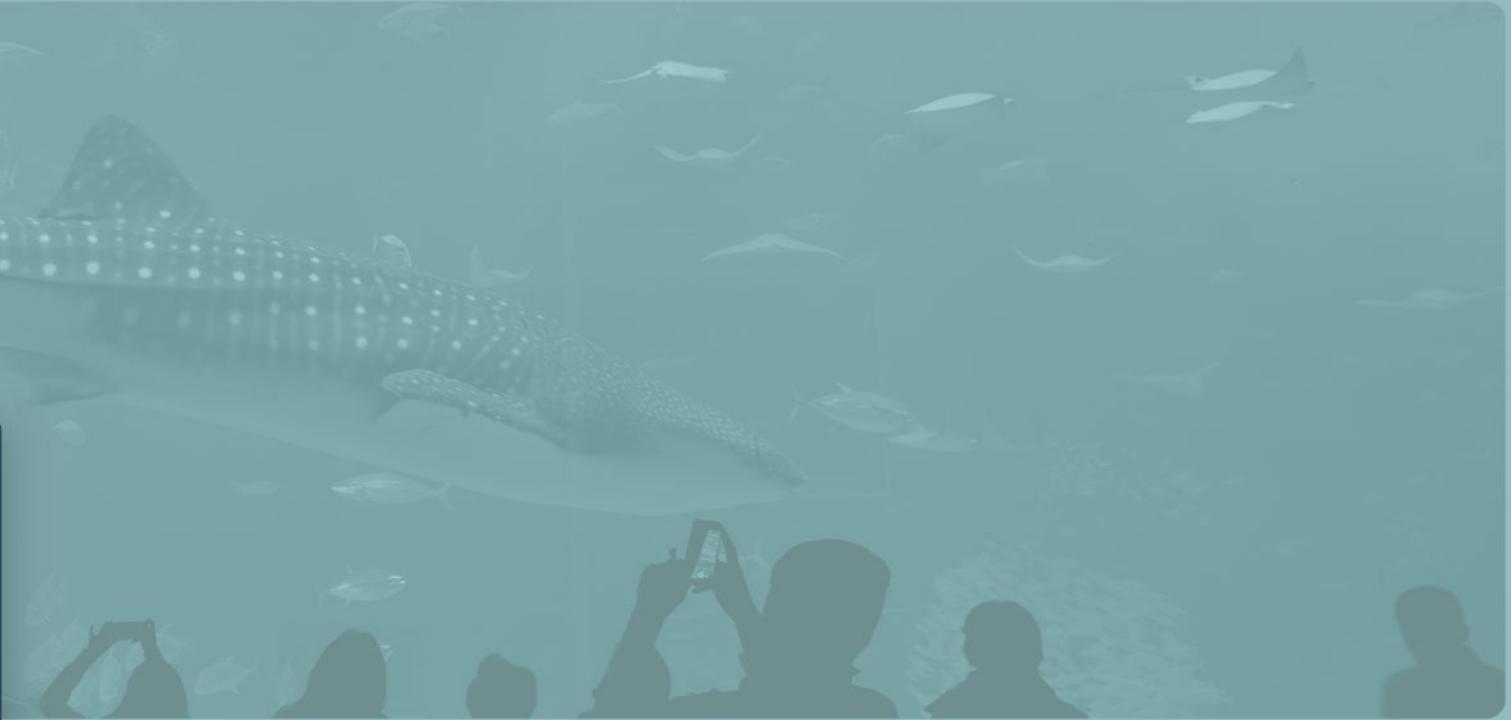
- Attach a commit as context to get a summary or explanation of the changes





Positron Assistant

Preview



To try Positron Assistant, enable the
`positron.assistant.enable` setting



positron.posit.co/assistant

Thank you :)

Other Positron-related sessions at conf:

- Talks in this room right after this by Austin Dickey and Wes McKinney!
- Multiple Console Sessions in Positron, Dhruvi Sompura
- Air - A blazingly fast R code formatter, Davis Vaughn, Lionel Henry
- How I got unstuck with Python, Julia Silge
- Positron Assistant Live Demo, George Stagg
- Databot Live Demo, Winston Chang



Stop by the **Positron Lounge** (Centennial Foyer) for a hands-on Positron Assistant demo, Q&As and more!