

Gli assistenti al codice IA

Vedi anche: [Introduzione agli strumenti IA per l'analisi dei dati](#)

Gli **assistenti IA integrati negli ambienti di sviluppo**, come *Codeion* (ora [Windsurf](#)), [GitHub Copilot](#), o [Gemini Code Assist](#) sono strumenti IA specializzati per il *coding*, addestrati su enormi quantità di codice *open source*, *repository* pubblici e documentazione tecnica. Durante questo processo di addestramento, l'IA impara a riconoscere pattern, strutture e convenzioni comuni nel codice di diversi linguaggi di programmazione.

La specializzazione e l'integrazione nell'IDE li rendono molto più efficienti rispetto ad una chat generica, ma non vi forniranno le previsioni del tempo o la ricetta della torta di mele 🤪.

Come funzionano

Una volta installati e/o impostati nell'IDE, essi leggono il file aperto (diciamo uno script di R) e cercano di “indovinare” cosa stiamo facendo sulla base di quello che stiamo scrivendo: offrono così, in tempo reale, **suggerimenti contestuali** su come completare il codice.

Il suggerimento appare come “testo fantasma”, cioè scritto in modo semi-invisibile: in genere, per accettare un suggerimento si preme il tasto Tab, mentre per ignorarlo è sufficiente continuare a scrivere (ed ecco perché si chiama “fantasma”).

È anche possibile usare **un commento come prompt** per descrivere ciò che si vuole fare, e l'assistente cercherà di generare il codice corrispondente. Ad esempio, scrivendo

```
# Calcola la media della variabile X
```

dopo l'invio, alla riga successiva, il modello suggerirà la funzione relativa, ovvero `mean(X)`.

Nell'**analisi dei dati**, gli usi tipici degli assistenti IA includono:

- Completamento automatico del codice;
- Completamento di attività ripetitive come i commenti;
- Suggerimenti di codice basati sul contesto;
- Generazione di codice ripetitivo, come cicli o funzioni;
- Esplorazione di nuove API e librerie.

Le chat integrate sono **agentive**, ovvero eseguono azioni complesse in autonomia: dalla creazione di uno script dal nulla, alla correzione di uno esistente, allo sviluppo di una applicazione. Le correzioni sono evidenziate una ad una, in maniera che l'utente possa accettarle/rifiutarle singolarmente o in blocco.

Anche se il completamento del codice o la creazione di blocchi di codice possono sembrare “manna dal cielo” — per chi non sa nulla del linguaggio che sta usando (o dell'IDE, persino), all'inizio possono creare parecchia **confusione**.

Oltre al fatto che, come già detto per le chat, non sempre i suggerimenti sono pertinenti o corretti, essi **“indovinano” su base probabilistica**: pertanto, in uno script vuoto, a fronte di un commento quale `# creo un vettore numerico`, l'assistente potrebbe suggerire `[1, 2, 3 ...]` (un vettore numerico in Python), invece che `c(1, 2, 3 ...)` (un vettore numerico in R). Python è infatti più usato di R, e quindi è il linguaggio più probabile.

I suggerimenti, inoltre, possono sovrapporsi all'autocompletamento già presente negli editor o ai suggerimenti dell'aiuto. E' un po' **come se Chat GPT rispondesse mentre stiamo scrivendo**, e questo all'inizio può essere complicato da gestire (si può scegliere di disattivare questa modalità per “chiamare” i suggerimenti con una combinazione di tasti).

[Questo video](#)  (brutto) mostra come funziona il completamento automatico, e la mia confusione, non simulata (Google Drive, si apre in altra scheda; ricordati di attivare i sottotitoli).

Con R: in RStudio e Positron

RStudio al momento non integra nessun assistente gratuito, ma solo GitHub Copilot, nelle versioni *Educational* e *Business*. La chat non è integrata.

Il nuovo IDE di Posit, **Positron** ha introdotto un **Positron Assistant**, progettato per interagire con lo stato attivo del progetto, offrendo suggerimenti contestuali, refactoring, debugging e assistenza “intelligente”. I modelli sono insomma integrati in maniera “profonda” nel contesto operativo e nell'interfaccia. Al momento si tratta *GitHub Copilot*, sempre nelle versioni *Educational* e *Business*, e *Claude*¹⁾.

Positron, però, è compatibile con VS Code e le sue estensioni, quindi permette di usare altri assistenti.

Per provare: gli strumenti gratuiti

Confronto di seguito i due assistenti IA più noti, **GitHub Copilot** e **Gemini Code Assist**, per quanto riguarda i limiti di utilizzo e le integrazioni disponibili in RStudio, Positron e VS Code.

GitHub Copilot

L'abbonamento *Educational* di **GitHub Copilot** è gratuito per gli studenti e gli insegnanti, e offre l'accesso completo a tutte le funzionalità, compresi i modelli avanzati (a scelta, nella chat, sono disponibili, oltre a GPT, anche Claude e Gemini).

Per ottenere l'abbonamento, è necessario creare un account GitHub con un indirizzo email scolastico e universitario, e poi seguire una procedura di verifica dell'appartenenza istituzionale che - al momento in cui scrivo - è alquanto complicata e invasiva della *privacy* (secondo me, ma credo anche secondo la nostra sensibilità di europei).

Per chi ha o *può ottenere* l'accesso a *GitHub Copilot Educational* è naturalmente l'opzione più comoda e accessibile.

Gemini Code Assist



In costruzione

Strumenti e IDE a Confronto

La selezione di seguito proposta si basa sulle capacità dei modelli di usare l'italiano nella comprensione e nella produzione dei testi (es.: i commenti), e sulla disponibilità di **piani gratuiti**, *non di prova e senza carta di credito*, per l'uso personale o didattico, che consentano di testarne le funzionalità senza dover investire in un abbonamento, e che *non richiedono la carta di credito*.

Ad uso e consumo, evidentemente di student* e insegnanti, nonché di principianti un po' più "avventuros*".

Tab. 1: Confronto fra GitHub Copilot e Gemini Code Assist, in RStudio e VS Code (settembre 2025)

Assistente	Limiti (gratuiti)	RStudio	Positron	VS Code
GitHub Copilot (account GitHub)	al mese: 2.000 completamenti di codice e 50 richieste per la chat / agente	<i>Educational / Business</i>	<i>Educational / Business</i>	<i>Personal (Free)</i>
Gemini Code Assist (account Google)	al giorno: 6.000 completamenti di codice e 240 richieste per la chat / agente	—	<i>Personal (Free)</i>	<i>Personal (Free)</i>

I 2.000 completamenti di codice al mese di Copilot sono chiaramente molti meno dei 6.000 al giorno di Gemini, ma si tratta comunque di un numero abbastanza elevato per un uso personale non intensivo.

Di contro, il numero di richieste per la chat è molto limitato (50 al mese contro le 240 al giorno di Gemini). Poiché all'inizio è probabile che la chat sarà molto più usata del completamento del codice, questo limite può essere raggiunto facilmente.

Usare RStudio e Positron (o altri) insieme

In ogni caso, per sperimentare con assistenti che non siano *GitHub Copilot*, è inevitabile provare un'altra IDE.

VS Code (*Visual Studio Code*; *open source* e gratuito) integra entrambi gli assistenti IA (e altri) nella versione *Personal / Free*.

Positron è costruito sulle componenti open source di VS Code, quindi è compatibile con le stesse estensioni. Essendo però un prodotto di Posit, e non di Microsoft, integra *GitHub Copilot* nelle stesse modalità di RStudio, e non gratuitamente. È però possibile usare, come visto nella tabella precedente, Gemini Code Assist, in un ambiente per alcuni aspetti simile a quello di RStudio.

Il *workflow* è abbastanza semplice in quanto i progetti vengono gestiti in modo simile (in cartelle dedicate), ed è quindi possibile passare da uno all'altro senza troppi problemi. Anzi, è possibile tenere aperto lo stesso progetto in entrambi gli IDE, facendo solo attenzione a salvare il file modificato in uno, prima di iniziare a lavorare nell'altro, per evitare conflitti e perdite di dati.

Usando RStudio come IDE primaria, è possibile lavorare sui file R in VS Code in due modi:

1. **Modo “semplice”:** Aprire un singolo script R (.R) con VS Code (File > Open File...) come quando si apre un file in un programma diverso da quello che lo ha generato.
2. **Buona pratica:** Aprire in VS Code l'intera **cartella del progetto** RStudio (File > Open Folder...), che verrà interpretata da VS Code (e dagli assistenti) per quello che è, cioè un progetto di analisi. Aprire la cartella è generalmente consigliato per progetti più strutturati, in quanto gli assistenti al codice interpretano tutti i file del progetto come “contesto”.

Aspetti etici dell'uso degli assistenti al codice

Gran parte del codice su cui questi modelli sono addestrati è open source. Il codice open source è generalmente rilasciato sotto una licenza specifica che impone di dichiararne l'uso e di distribuire le modifiche o le opere derivate anch'esse in formato aperto. È fondamentale rispettare queste licenze quando si utilizza codice suggerito da un *code assistant*, specialmente in progetti complessi, destinati alla distribuzione pubblica.

Molti assistenti IA, soprattutto quelli progettati per la generazione di codice, sono ora in grado di integrare (parzialmente) riferimenti e citazioni quando necessario. Questo è particolarmente vero quando il codice generato si basa su librerie, framework o esempi specifici. Copertura e affidabilità delle citazioni non è però ancora garantita al 100%.

[RStudio](#), [ide](#), [IA](#), [coding](#)

1)

Manco forse a specificarlo, ma gli abbonamenti ai servizi di chat — ad esempio Claude plus — non forniscono l'accesso via API.

From:

<https://www.agnesevardanega.eu/wiki/> - **Ricerca Sociale con R**

Permanent link:

<https://www.agnesevardanega.eu/wiki/r/rstudio/ai-code-assistant>

Last update: **15/10/2025 07:17**

