

AI tools in programming

Scientific workflows: Tools and Tips 

Dr. Selina Baldauf

2024-01-18

What is this lecture series?

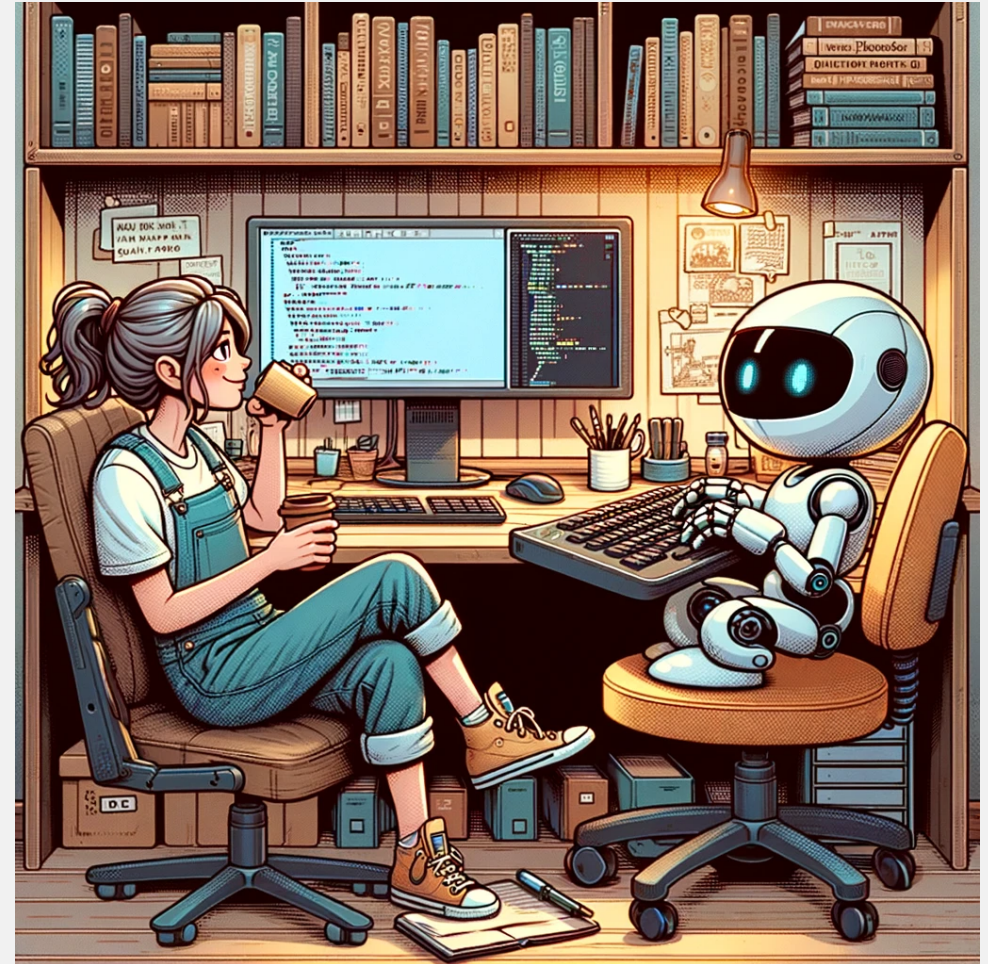
Scientific workflows: Tools and Tips

 Every 3rd Thursday  4-5 p.m.  Webex

- One topic from the world of scientific workflows
- Material provided [online](#)
- If you don't want to miss a lecture
 - [Subscribe to the mailing list](#)

Motivation

- AI tools assist programmers with
 - Coding
 - Debugging
 - Learning
 - ...
- Higher productivity and efficiency
- More motivation



Overview of tools

- Browser-based chat bots ([ChatGPT](#), [Bard](#), ...)
 - General-purpose
- Data-analysis tools (Data analyst GPT, [RTutor](#), ...)
 - Upload data and ask questions about it
 - Download the code that was used for the results
- Integrated AI tools ([GitHub Copilot](#), [Codium AI](#), ...)
 - Integrated directly in programming environment
 - Real-time suggestions, chat, debugging, ...

Today

- Focus on **integrated AI tools**
 - How to use GitHub Copilot to
 - Speed up your coding
 - Improve your code
 - Learn
- Concerns when using AI tools
- **Main goal:** Motivate you to try out tools and find out what fits your workflow
- Find other tools on the [website](#)

Now You

- ? What is your main programming language
- ? Which IDE (programming environment) do you use
- ? Which AI tools for programming did you already try

Integrated AI tools for programming

Mainly GitHub Copilot

GitHub Copilot

- Cloud-based AI tool by Github and OpenAI
- Model based on GPT-4 and OpenAI's Codex
 - Specifically trained on source code
- Basic idea: Plugin for your IDE to integrate Copilot
- Works best for well-represented languages (Python, JS, ...)

How to get GitHub Copilot

See [lecture website](#) for step-by-step guide and more information.

It's really easy, but you need:

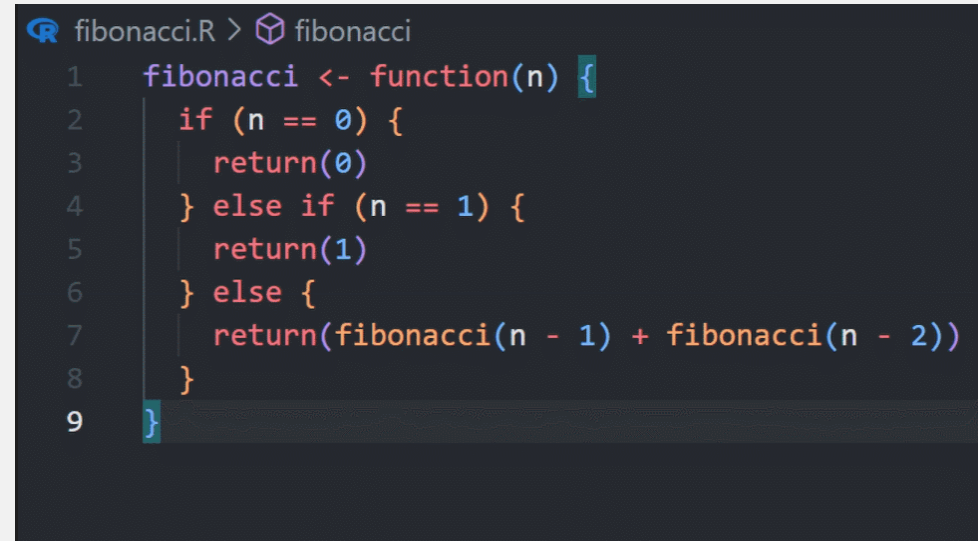
- GitHub Account
- Active GH Copilot subscription (10\$ per month)
 - Get it for free as an academic with an educational account
- IDE that supports Copilot
 - Full support: Visual Studio (Code), Vim, Neovim, JetBrains IDEs (e.g. PyCharm)
 - Limited support: RStudio, ?

Using GitHub Copilot

Demo of the main features and use cases

Inline code suggestions

- Copilot tries to predict what you want to do next
- Suggestions are based on the context
 - Previous code
 - Comments
 - Variable and function names
 - ...



```
fibonacci.R > fibonacci
1  fibonacci <- function(n) {
2    if (n == 0) {
3      return(0)
4    } else if (n == 1) {
5      return(1)
6    } else {
7      return(fibonacci(n - 1) + fibonacci(n - 2))
8    }
9  }
```

Get better suggestions

- **Provide context**
 - Open other files
 - Add top level comments explaining the purpose of the script
 - Name variables and functions properly
 - Copy-paste sample code and delete it later
- **Be consistent**
 - “Garbage in, garbage out”
 - Have a nice and consistent coding style

Nice side effect of using Copilot: More good-practice coding

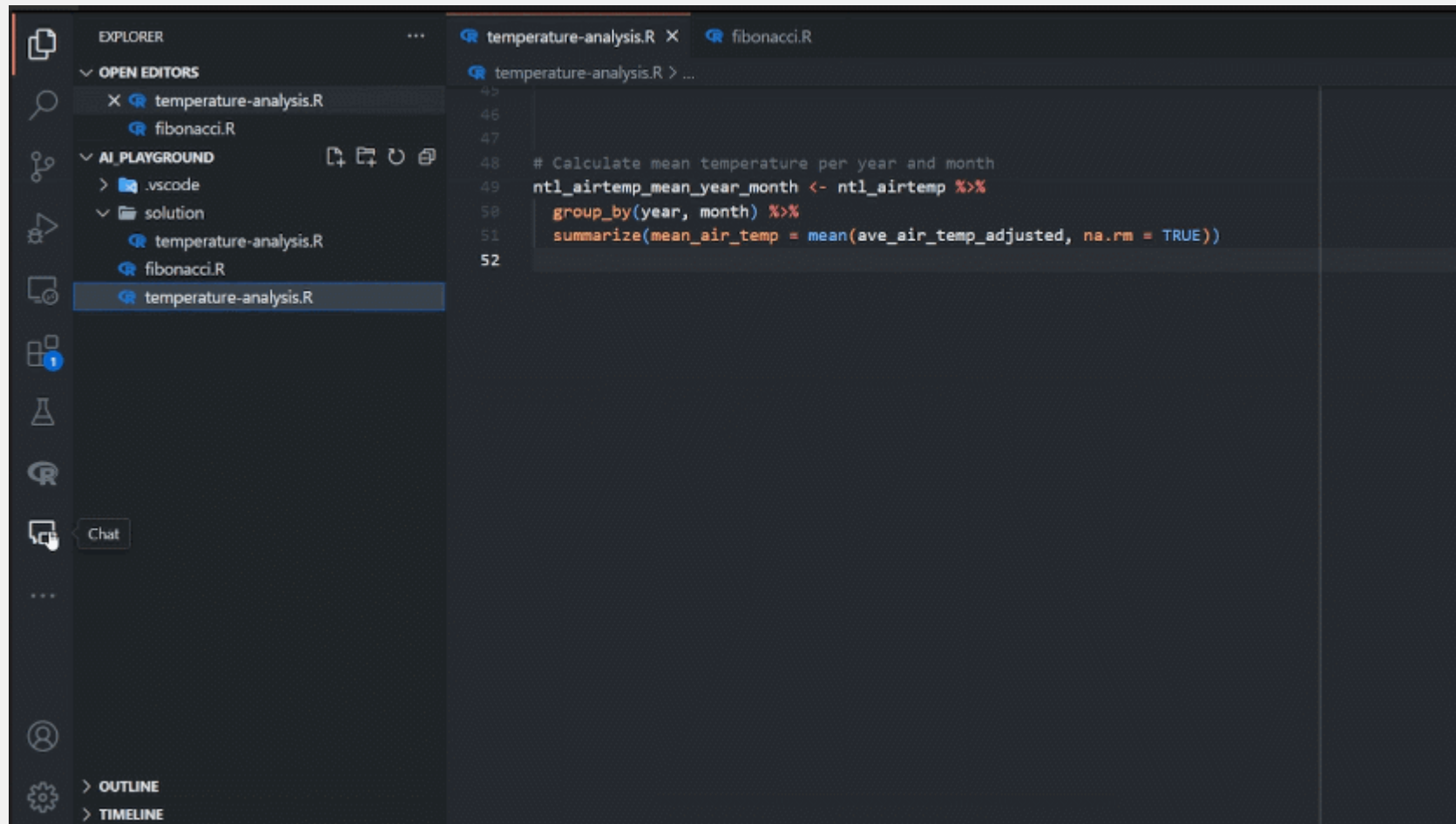
Chat

- Ask and give commands regarding:
 - Highlighted lines of code
 - The whole script or project
- Preset commands starting with /
 - `/fix`: fix problems in your code
 - `/doc`: get documentation
 - `/explain`: explain this code
 - `/test`: write unit tests
 - `/new`: create new projects or scripts with code

/fix with in-line chat

```
fibonacci.R > fibonacci
1 fibonacci <- function(n) {
2   if (n == 0) {
3     return(0)
4   } else if (n == 1) {
5     return(1)
6   } else {
7     return(fibonacci(n - 1) + fibonacci(n - 2))
8   }
9 }
10
```

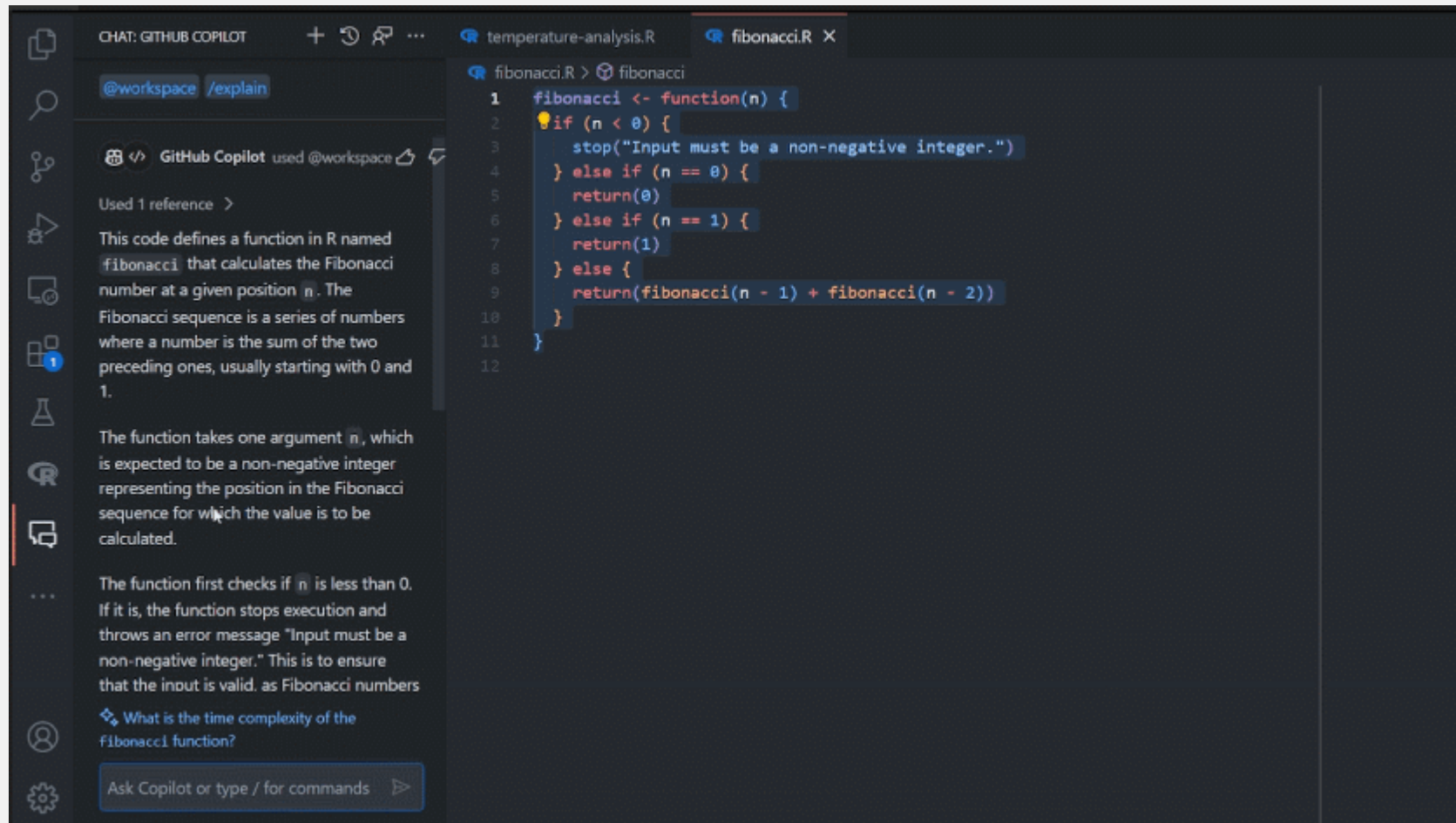
/fix with chat in the sidebar



/doc with specific documentation standard

```
fibonacci.R > fibonacci
1  #' Calculate the nth Fibonacci number.
2  #'
3  #' This function calculates the nth Fibonacci number using recursion.
4  #'
5  #' @param n The position of the Fibonacci number to calculate.
6  #' @return The nth Fibonacci number.
7  #' @examples
8  #' fibonacci(0)
9  #' fibonacci(1)
10 #' fibonacci(5)
11 #' @export
12 fibonacci <- function(n) {
13   if (n == 0) {
14     return(0)
15   } else if (n == 1) {
16     return(1)
17   } else {
18     return(fibonacci(n - 1) + fibonacci(n - 2))
19   }
20 }
```

/explain



The screenshot displays the GitHub Copilot interface within a code editor. On the left, a chat window titled "CHAT: GITHUB COPILOT" shows a conversation. The user has entered the command `@workspace /explain`. The Copilot response explains the `fibonacci` function in R, detailing its purpose, arguments, and logic. The right pane shows the R code for the `fibonacci` function, which includes input validation and recursive calls.

Chat Window:

CHAT: GITHUB COPILOT

`@workspace /explain`

Used 1 reference >

This code defines a function in R named `fibonacci` that calculates the Fibonacci number at a given position `n`. The Fibonacci sequence is a series of numbers where a number is the sum of the two preceding ones, usually starting with 0 and 1.

The function takes one argument `n`, which is expected to be a non-negative integer representing the position in the Fibonacci sequence for which the value is to be calculated.

The function first checks if `n` is less than 0. If it is, the function stops execution and throws an error message "Input must be a non-negative integer." This is to ensure that the input is valid, as Fibonacci numbers

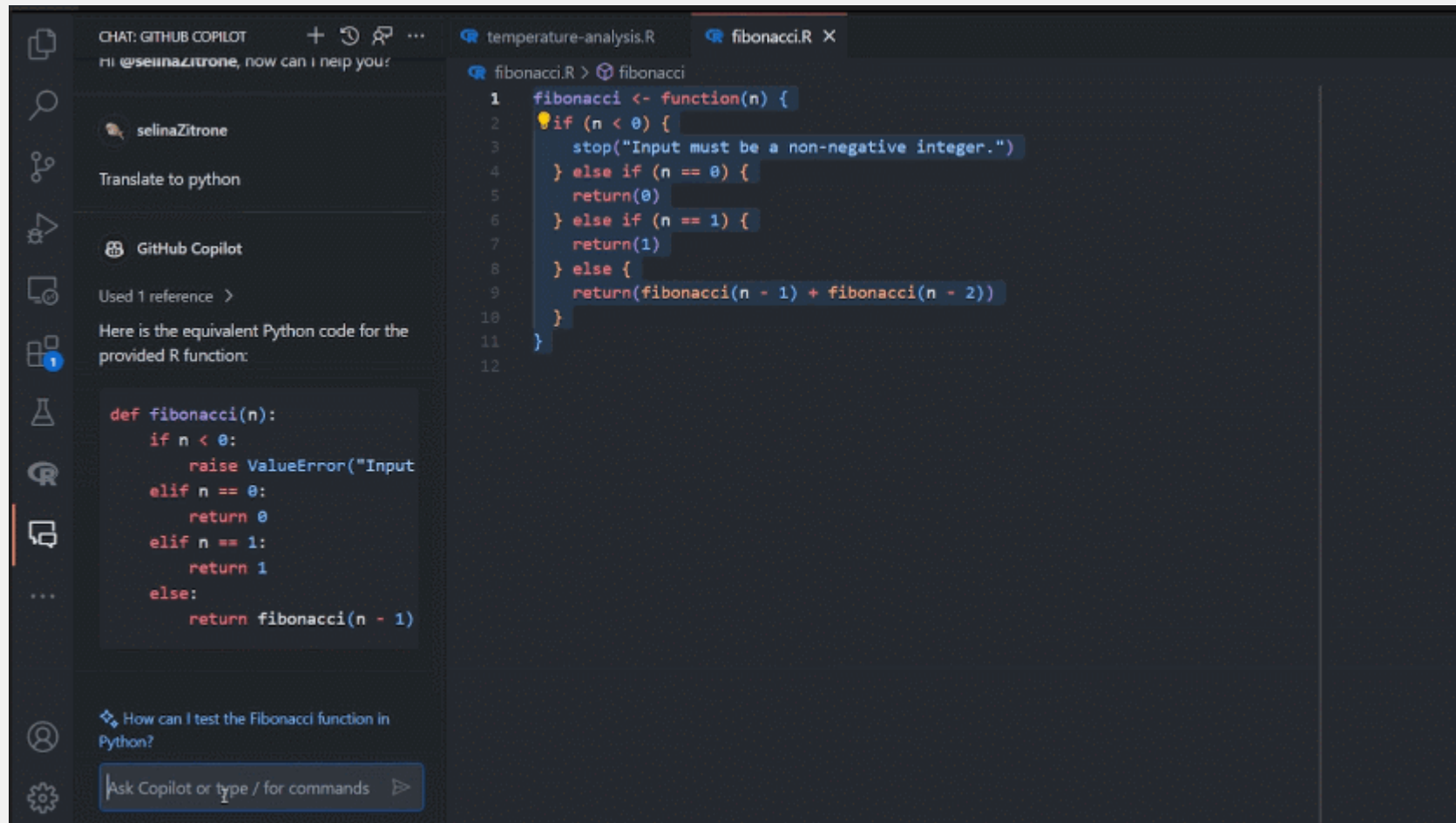
What is the time complexity of the `fibonacci` function?

Ask Copilot or type / for commands

Code Editor:

```
1 fibonacci <- function(n) {  
2   if (n < 0) {  
3     stop("Input must be a non-negative integer.")  
4   } else if (n == 0) {  
5     return(0)  
6   } else if (n == 1) {  
7     return(1)  
8   } else {  
9     return(fibonacci(n - 1) + fibonacci(n - 2))  
10  }  
11 }  
12
```

Translate code



Codium AI as an alternative

- No inline code suggestions
- Great functionality to
 - Explain code
 - Suggestions improve and enhance code
 - Generate tests
- Not in RStudio, but in VS code and many other IDEs
- Free for personal use (for now)

Concerns to consider

- Privacy
 - Chose whether your prompts and suggestions will be used by Github (Github -> Seetings -> Copilot -> Policies)
- Plagiarism
 - Block suggestions matching public code (Github -> Seetings -> Copilot -> Policies)
- Ethical concerns
 - For-profit tool trained on open-source
- Environmental concerns
 - Water and enery usage

Usage guidelines

- No definite guidelines, but see examples on [lecture website](#)
- **Responsibility**
 - You are responsible for your scientific output
 - Stay critical, double-check
- **Transparency**
 - Make clear for which tasks you used which AI
- **Know relevant guidelines**
 - Journals
 - Your university
- Don't use AI in exams

Summary

- AI tools for programming can be extremely useful
- Try different tools and find the ones you like
- Think about concerns
- Learn about relevant guidelines
- Development is fast, so keep up
- Check out the [lecture website](#) if you want to get started

Next lecture

Topic t.b.a.

 15th February  4-5 p.m.  Webex

 [Subscribe to the mailing list](#)

 For topic suggestions and/or feedback [send me an email](#)

Thank you for your attention :)

Questions?

Thanks to Anne Lewerentz for support with the preparation.

References

- Experiment on programmer efficiency with AI tools
- GitHub Copilot
- GitHub Copilot privacy FAQ
- [GitHub Copilot Docs](#): Useful information and guides on how to use Copilot
- Prompt engineering with GitHub Copilot
- Codium AI

Guidelines

- DFG Rules on the use of AI particularly for proposals
- Nature living guidelines on responsible use of generative AI in research
- EU AI Act
- Universities (German)
 - FU Berlin “Eckpunktepapier” (German)
 - TU Berlin on AI: Mainly about AI in teaching but contains some general links to other guidelines

