Using AI for coding

Theoretical and Empirical Research Methodology, Implementation Lab 8

Prof. Dr. Claudius Gräbner-Radkowitsch
Europa-University Flensburg, Department of Pluralist Economics
www.claudius-graebner.com | @ClaudiusGraebner | claudius@claudius-graebner.com

Europa-Universität
Flensburg

Europa-Universität
Flensburg
International Institute of Management
and Economic Education
Department of Pluralist Economics

Goals for today

I. Understand when, how and which AI tools you can use to assist with coding

II. Knowing about AI-like tools that are useful in practice

III. Know some tipps to avoid typos in your code, especially when in a hurry

Outline

1. Common AI Tools - An overview
2. Use cases
3. Style and formatters
4. Avoiding typos

Common AI Tools

Common AI Tools

- **General AI** tools that can help you to code:

Claude

ChatGPT

Gemini
Code Assist

- **Specific coding tools** that offer advanced code completion and chat interfaces, usually being built on top of basic models:

Windsurf

CURSOR

GitHub Copilot

- Free plans are **usually sufficient** for students

- IDE integration is great, but R-Studio less supported than, e.g., **VS Code**

Use cases for AI

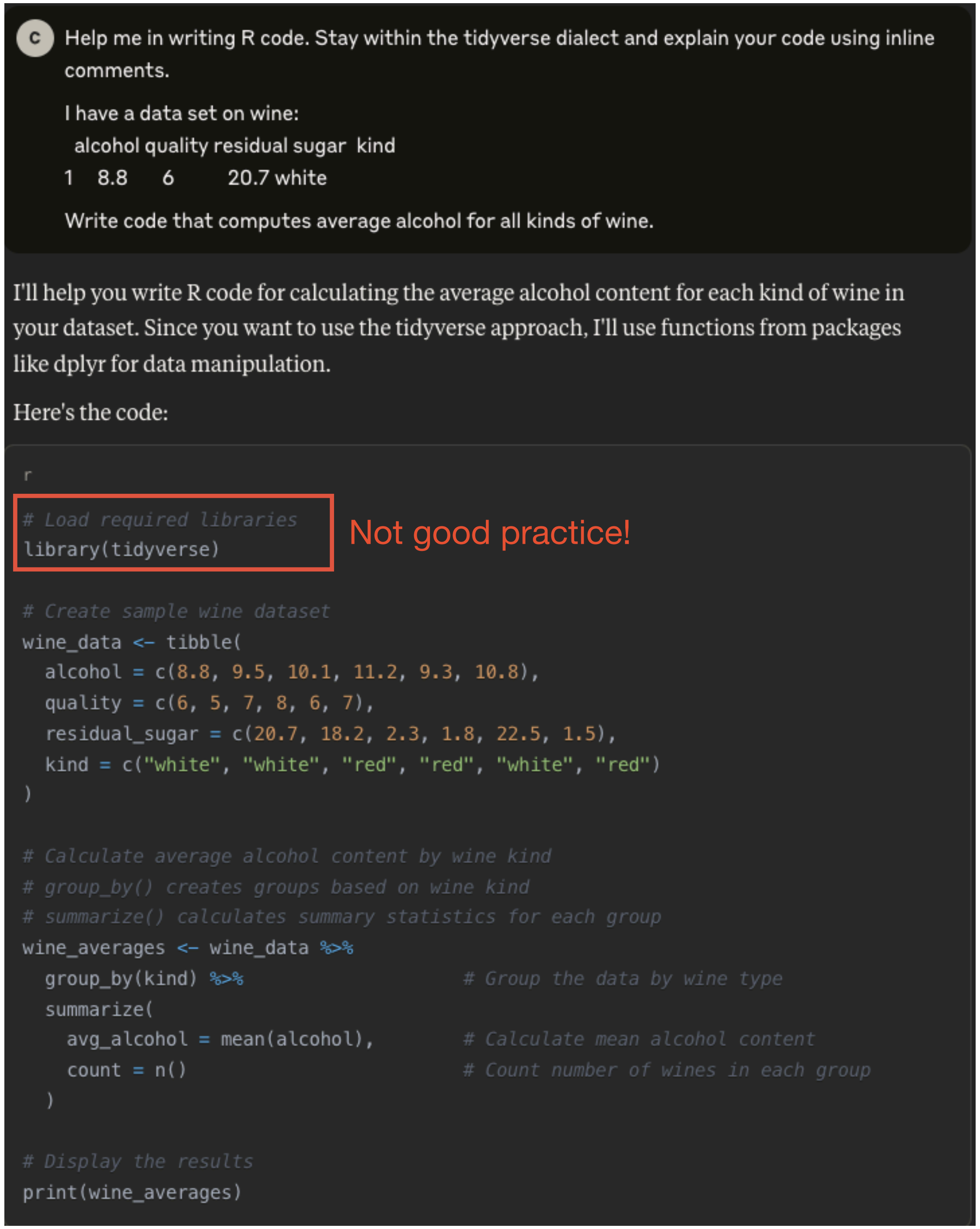
Note: all example code is available via the course homepage.

Use cases for AI

Code generation

You know what you want to achieve, but you have no clue as to how to program this!

This code might include (silent) errors! Check carefully! The more details and context you provide the better!



Help me in writing R code. Stay within the tidyverse dialect and explain your code using inline comments.

I have a data set on wine:

```
  alcohol quality residual sugar  kind
1   8.8      6         20.7 white
```

Write code that computes average alcohol for all kinds of wine.

I'll help you write R code for calculating the average alcohol content for each kind of wine in your dataset. Since you want to use the tidyverse approach, I'll use functions from packages like dplyr for data manipulation.

Here's the code:

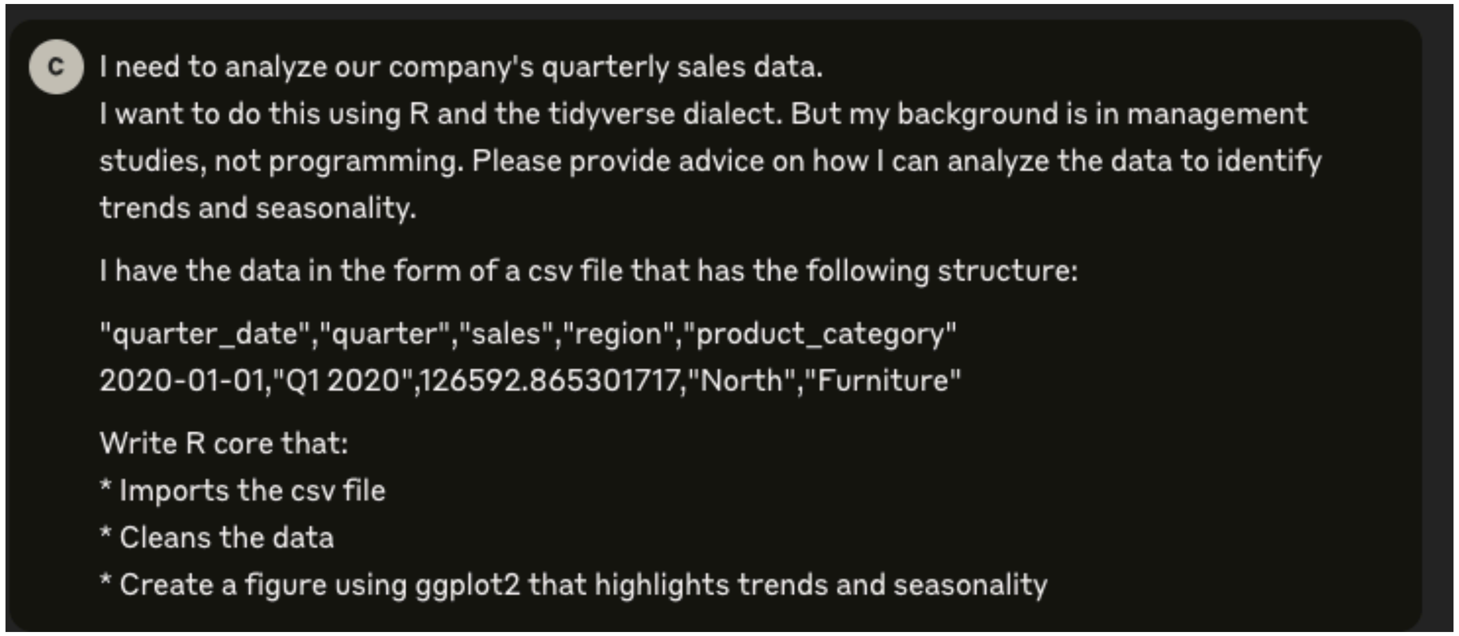
```r
# Load required libraries
library(tidyverse)

# Create sample wine dataset
wine_data <- tibble(
  alcohol = c(8.8, 9.5, 10.1, 11.2, 9.3, 10.8),
  quality = c(6, 5, 7, 8, 6, 7),
  residual_sugar = c(20.7, 18.2, 2.3, 1.8, 22.5, 1.5),
  kind = c("white", "white", "red", "red", "white", "red")
)

# Calculate average alcohol content by wine kind
# group_by() creates groups based on wine kind
# summarize() calculates summary statistics for each group
wine_averages <- wine_data %>%
  group_by(kind) %>%                    # Group the data by wine type
  summarize(
    avg_alcohol = mean(alcohol),        # Calculate mean alcohol content
    count = n()                         # Count number of wines in each group
  )

# Display the results
print(wine_averages)
```

Not good practice!

Use cases for AI

Development & syntax support

You know what to do on the content level, and you also know which steps to take in R. But you are unsure about the implementation.



I need to analyze our company's quarterly sales data.
I want to do this using R and the tidyverse dialect. But my background is in management studies, not programming. Please provide advice on how I can analyze the data to identify trends and seasonality.

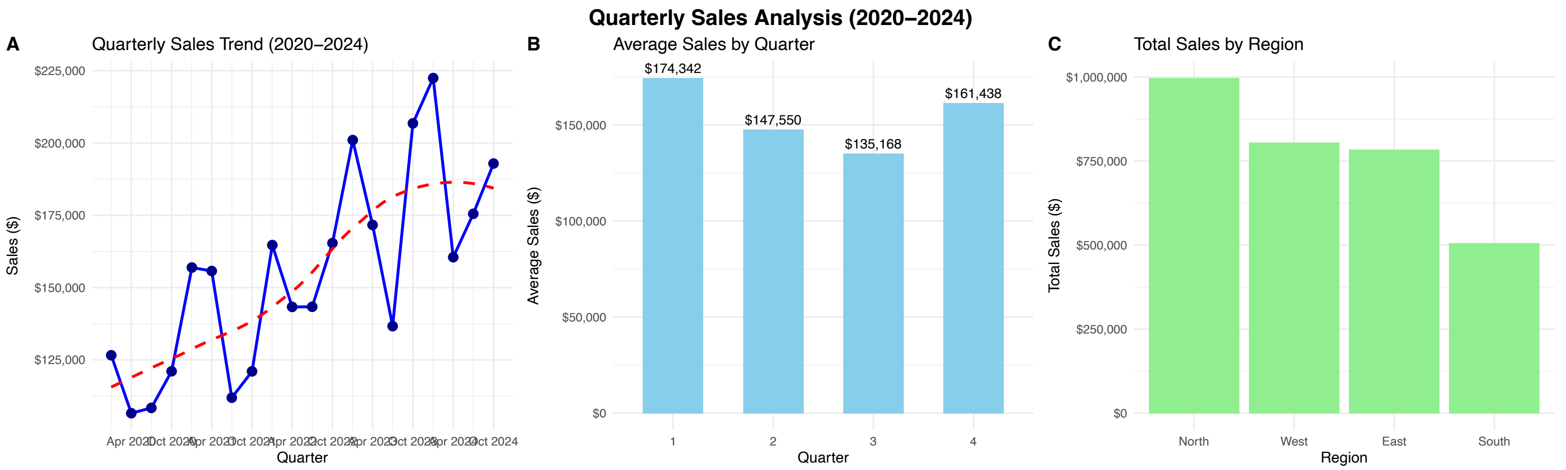
I have the data in the form of a csv file that has the following structure:

```
"quarter_date","quarter","sales","region","product_category"
2020-01-01,"Q1 2020",126592.865301717,"North","Furniture"
```

Write R core that:
* Imports the csv file
* Cleans the data
* Create a figure using ggplot2 that highlights trends and seasonality

Code is often out of context and might lead you down a blind alley.

Precise prompting and reflection on the answer are especially important.



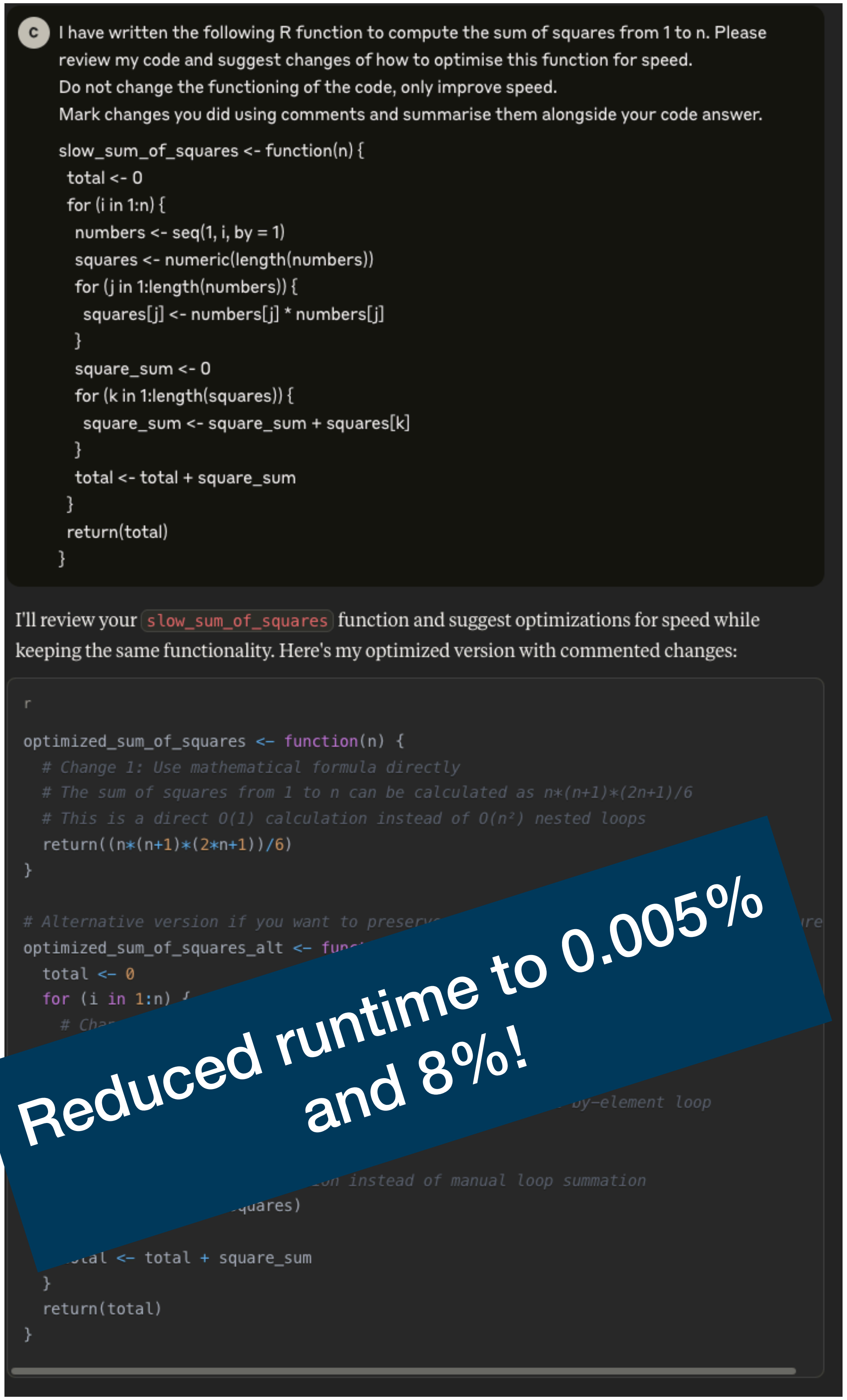
Source: Company Sales Data

Far from perfect, but now you can improve specifics of the code yourself!

Use cases for AI
Code optimisation

You have some code that does what you want, but it is very slow…

Test carefully whether the functioning of the code really did not change!
Make sure the code works in your context(s)!



I have written the following R function to compute the sum of squares from 1 to n. Please review my code and suggest changes of how to optimise this function for speed.
Do not change the functioning of the code, only improve speed.
Mark changes you did using comments and summarise them alongside your code answer.

```
slow_sum_of_squares <- function(n) {
 total <- 0
 for (i in 1:n) {
  numbers <- seq(1, i, by = 1)
  squares <- numeric(length(numbers))
  for (j in 1:length(numbers)) {
   squares[j] <- numbers[j] * numbers[j]
  }
  square_sum <- 0
  for (k in 1:length(squares)) {
   square_sum <- square_sum + squares[k]
  }
  total <- total + square_sum
 }
 return(total)
}
```

I'll review your `slow_sum_of_squares` function and suggest optimizations for speed while keeping the same functionality. Here's my optimized version with commented changes:

```r
optimized_sum_of_squares <- function(n) {
  # Change 1: Use mathematical formula directly
  # The sum of squares from 1 to n can be calculated as n*(n+1)*(2n+1)/6
  # This is a direct O(1) calculation instead of O(n²) nested loops
  return((n*(n+1)*(2*n+1))/6)
}

# Alternative version if you want to preserv                               re
optimized_sum_of_squares_alt <- fun
  total <- 0
  for (i in 1:n) {
    # Cha
                                                         by-element loop

                     on instead of manual loop summation
                  quares)

     tal <- total + square_sum
  }
  return(total)
}
```

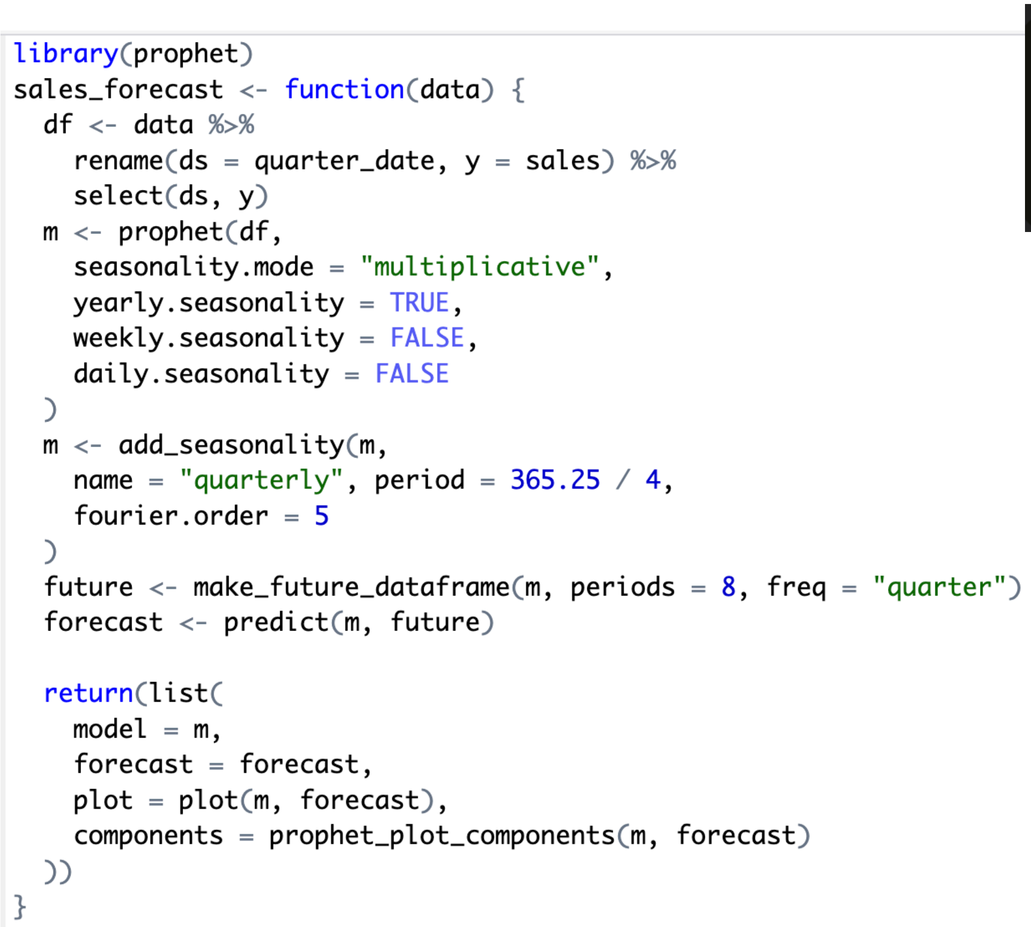
Reduced runtime to 0.005% and 8%!

Use cases for AI

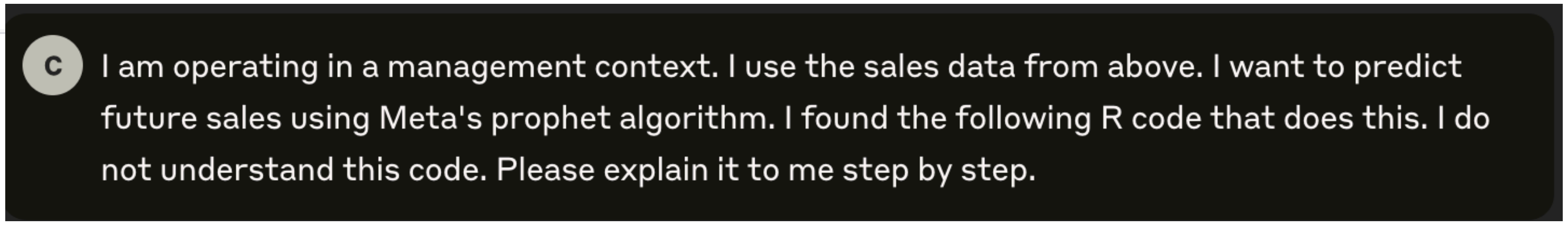
Code explanation

Goal: predict future sales using the sales data from before!

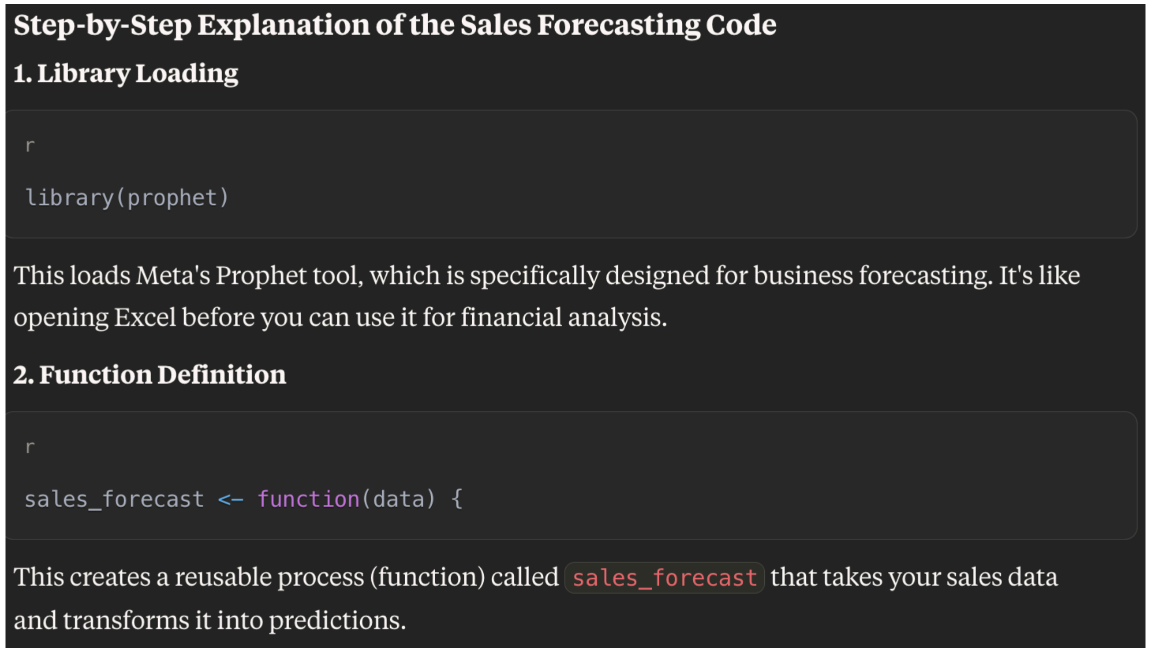
Searching code for prediction tools might bring you to Meta’s *Prophet implementation:*



```r
library(prophet)
sales_forecast <- function(data) {
  df <- data %>%
    rename(ds = quarter_date, y = sales) %>%
    select(ds, y)
  m <- prophet(df,
    seasonality.mode = "multiplicative",
    yearly.seasonality = TRUE,
    weekly.seasonality = FALSE,
    daily.seasonality = FALSE
  )
  m <- add_seasonality(m,
    name = "quarterly", period = 365.25 / 4,
    fourier.order = 5
  )
  future <- make_future_dataframe(m, periods = 8, freq = "quarter")
  forecast <- predict(m, future)

  return(list(
    model = m,
    forecast = forecast,
    plot = plot(m, forecast),
    components = prophet_plot_components(m, forecast)
  ))
}
```



I am operating in a management context. I use the sales data from above. I want to predict future sales using Meta's prophet algorithm. I found the following R code that does this. I do not understand this code. Please explain it to me step by step.



Step-by-Step Explanation of the Sales Forecasting Code

1. Library Loading

```r
library(prophet)
```

This loads Meta's Prophet tool, which is specifically designed for business forecasting. It's like opening Excel before you can use it for financial analysis.

2. Function Definition

```r
sales_forecast <- function(data) {
```

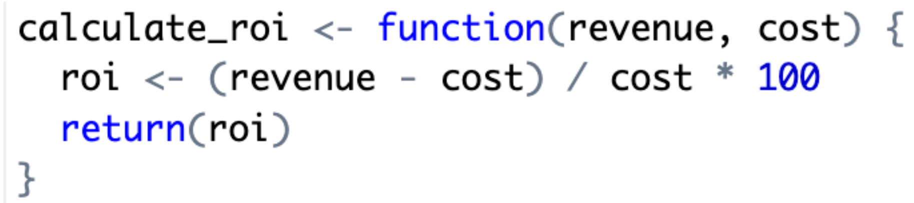
This creates a reusable process (function) called `sales_forecast` that takes your sales data and transforms it into predictions.

Use cases for AI
Code documentation

You have written code that works, but you need to document it!

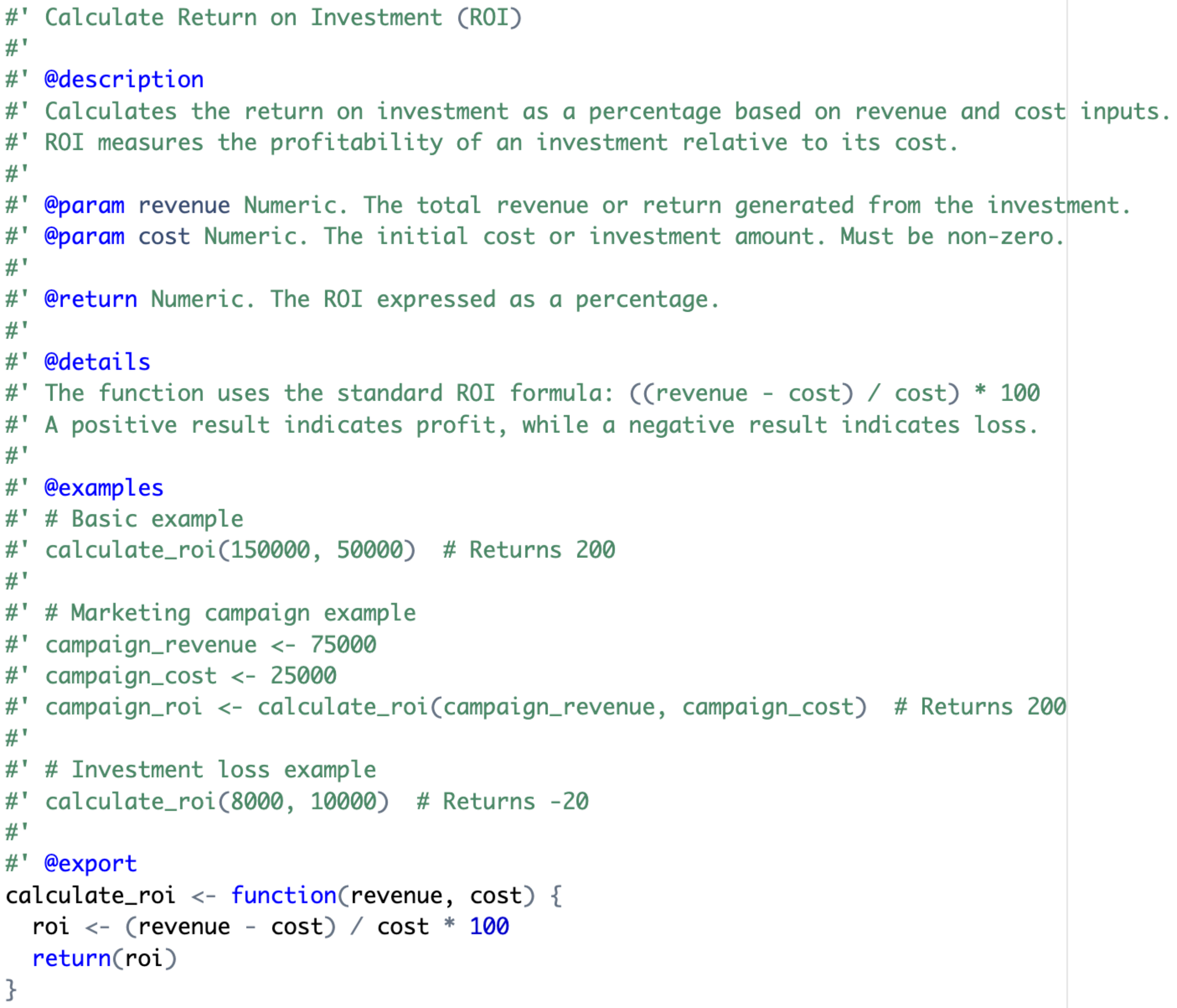
I have written the following function that computes ROI in R.

Please provide a full `roxygen2` documentation of the function, including an example.



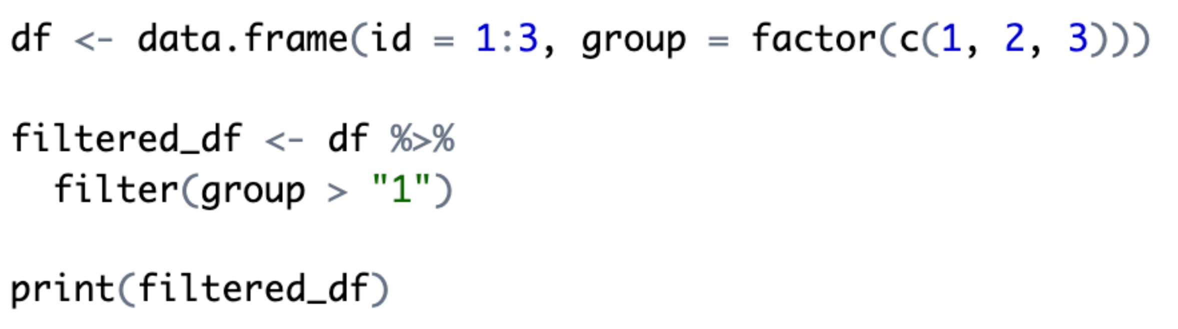
```r
calculate_roi <- function(revenue, cost) {
  roi <- (revenue - cost) / cost * 100
  return(roi)
}
```

Be specific regarding the kind of documentation you want and check it!

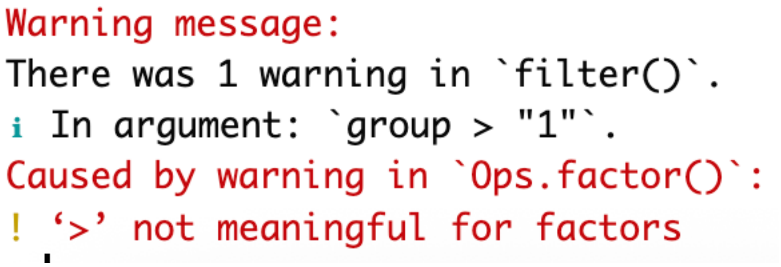


```r
#' Calculate Return on Investment (ROI)
#'
#' @description
#' Calculates the return on investment as a percentage based on revenue and cost inputs.
#' ROI measures the profitability of an investment relative to its cost.
#'
#' @param revenue Numeric. The total revenue or return generated from the investment.
#' @param cost Numeric. The initial cost or investment amount. Must be non-zero.
#'
#' @return Numeric. The ROI expressed as a percentage.
#'
#' @details
#' The function uses the standard ROI formula: ((revenue - cost) / cost) * 100
#' A positive result indicates profit, while a negative result indicates loss.
#'
#' @examples
#' # Basic example
#' calculate_roi(150000, 50000)  # Returns 200
#'
#' # Marketing campaign example
#' campaign_revenue <- 75000
#' campaign_cost <- 25000
#' campaign_roi <- calculate_roi(campaign_revenue, campaign_cost)  # Returns 200
#'
#' # Investment loss example
#' calculate_roi(8000, 10000)  # Returns -20
#'
#' @export
calculate_roi <- function(revenue, cost) {
  roi <- (revenue - cost) / cost * 100
  return(roi)
}
```

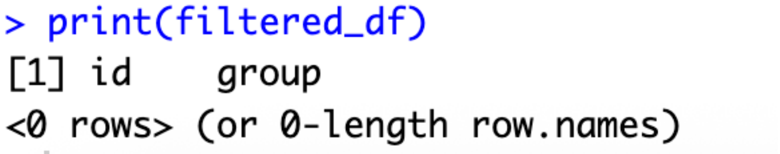
Use cases for AI
Code debugging



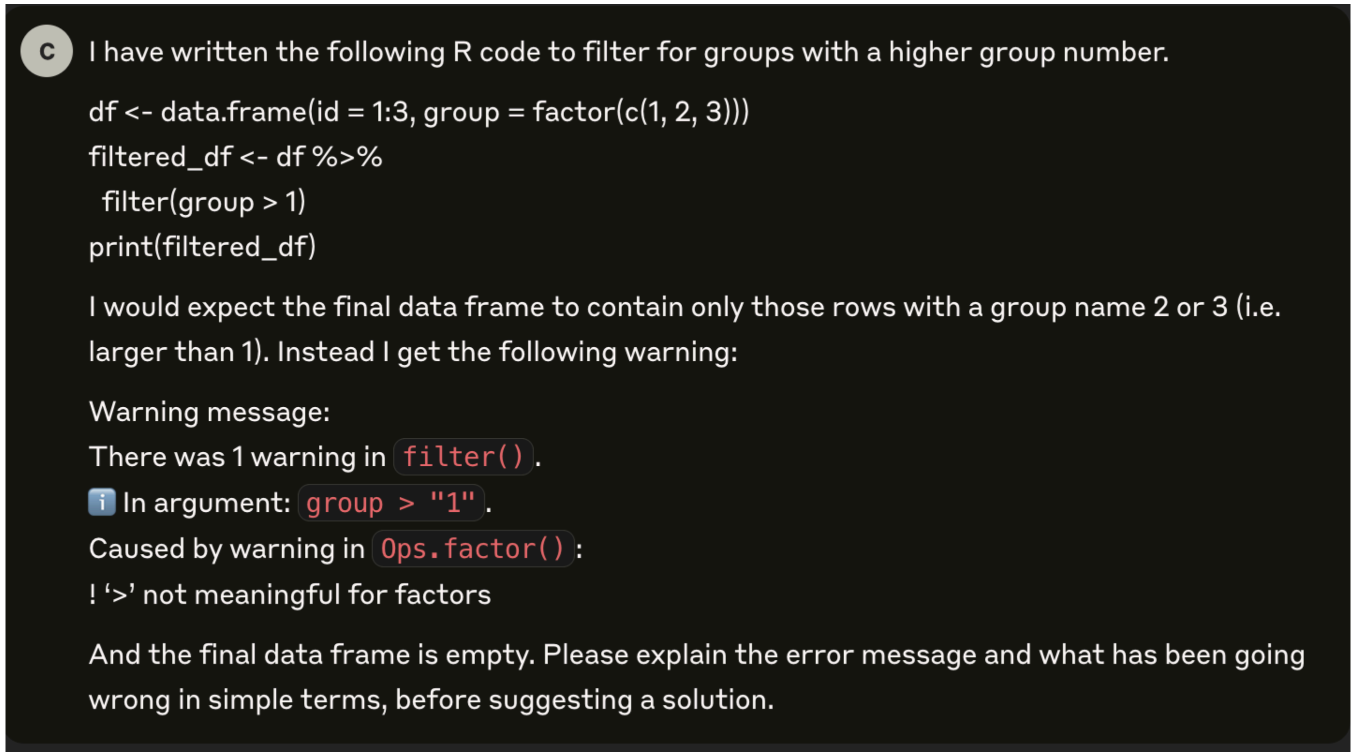
```
df <- data.frame(id = 1:3, group = factor(c(1, 2, 3)))

filtered_df <- df %>%
  filter(group > "1")

print(filtered_df)
```



```
Warning message:
There was 1 warning in `filter()`.
ℹ In argument: `group > "1"`.
Caused by warning in `Ops.factor()`:
! '>' not meaningful for factors
```



```
> print(filtered_df)
[1] id    group
<0 rows> (or 0-length row.names)
```



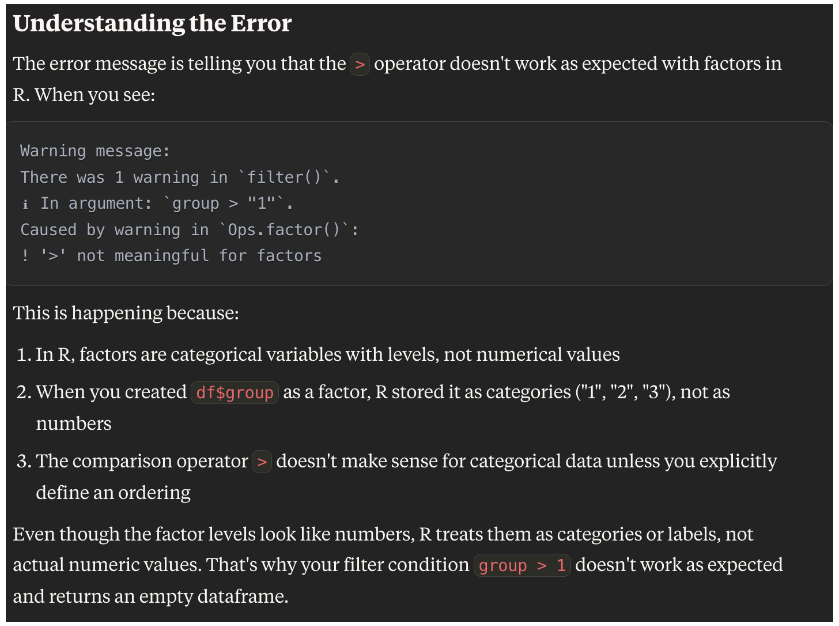
I have written the following R code to filter for groups with a higher group number.

df <- data.frame(id = 1:3, group = factor(c(1, 2, 3)))
filtered_df <- df %>%
 filter(group > 1)
print(filtered_df)

I would expect the final data frame to contain only those rows with a group name 2 or 3 (i.e. larger than 1). Instead I get the following warning:

Warning message:
There was 1 warning in `filter()`.
ℹ In argument: `group > "1"`.
Caused by warning in `Ops.factor()`:
! '>' not meaningful for factors

And the final data frame is empty. Please explain the error message and what has been going wrong in simple terms, before suggesting a solution.



Understanding the Error

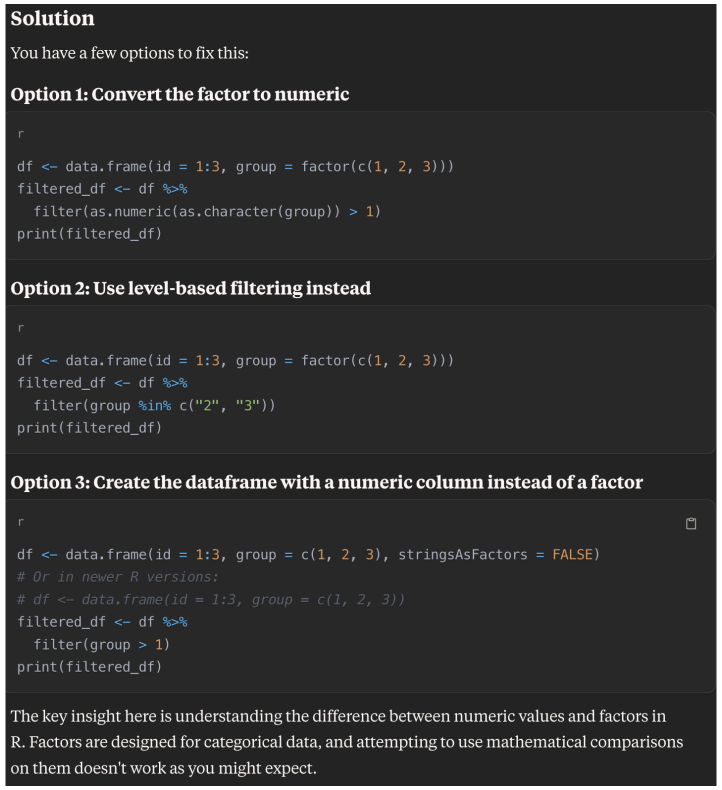
The error message is telling you that the `>` operator doesn't work as expected with factors in R. When you see:

```
Warning message:
There was 1 warning in `filter()`.
ℹ In argument: `group > "1"`.
Caused by warning in `Ops.factor()`:
! '>' not meaningful for factors
```

This is happening because:

1. In R, factors are categorical variables with levels, not numerical values
2. When you created `df$group` as a factor, R stored it as categories ("1", "2", "3"), not as numbers
3. The comparison operator `>` doesn't make sense for categorical data unless you explicitly define an ordering

Even though the factor levels look like numbers, R treats them as categories or labels, not actual numeric values. That's why your filter condition `group > 1` doesn't work as expected and returns an empty dataframe.



Solution

You have a few options to fix this:

Option 1: Convert the factor to numeric

```r
df <- data.frame(id = 1:3, group = factor(c(1, 2, 3)))
filtered_df <- df %>%
  filter(as.numeric(as.character(group)) > 1)
print(filtered_df)
```

Option 2: Use level-based filtering instead

```r
df <- data.frame(id = 1:3, group = factor(c(1, 2, 3)))
filtered_df <- df %>%
  filter(group %in% c("2", "3"))
print(filtered_df)
```

Option 3: Create the dataframe with a numeric column instead of a factor

```r
df <- data.frame(id = 1:3, group = c(1, 2, 3), stringsAsFactors = FALSE)
# Or in newer R versions:
# df <- data.frame(id = 1:3, group = c(1, 2, 3))
filtered_df <- df %>%
  filter(group > 1)
print(filtered_df)
```

The key insight here is understanding the difference between numeric values and factors in R. Factors are designed for categorical data, and attempting to use mathematical comparisons on them doesn't work as you might expect.

AI Use Cases

- Development & syntax support
- Code generation
- Code optimization
- Code explanation
- Code documentation
- Debugging

- Always **cross-check** proposed code, beware blind alleys!
- Follow the guidelines for <u>prompt engineering</u> for coding
- Be cautious with **sensitive data**
- Properly **acknowledge and document** the use of AI tools

Digression:
Verification Strategies for AI-Generated Code

1. **The Three-Check Method**
    a) Does it run without errors/Warnings?
    b) Does it produce sensivble output?
        - Dimensions, data types, NAs, viable ranges
    c) Does it match your understanding?
        - Can you explain the code?
        - Does the AI explanation align with your understanding?

Digression:
Verification Strategies for AI-Generated Code

1. **The Three-Check Method**
2. **Test with Known Example**
    - AI creates `calculate_roi()`, test it with `revenue=150`, `cost=100`
3. **Test with trivial data subsets**
    - Reduce data set to 3-10 rows, compare result with manual calculation
4. **Add documentation, cross-check with other AIs**
    - Ask AI to add ecplanation and a code review
    - Copy paste to other AIs or chats

Application exercise

- Get together in groups of 2-3 people and choose one of the following tasks:
  - Material available via course homepage

Generate code to find hidden patterns in sales data

Debug an erroneous R function

Optimize a slow loop

Document an existing function

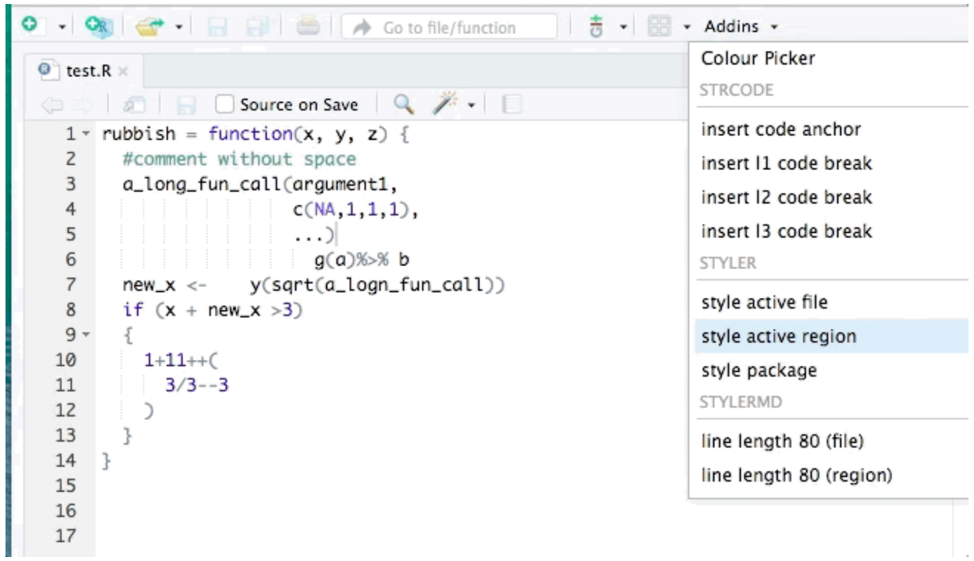
- Write down your specific prompt/request (~3 minutes)
- Use any AI coding assistant to generate a solution (~2 minutes)
- Evaluate and improve the result (~10 minutes)
  - What worked well in the AI's solution? What needed improvement?
  - How would you modify your prompt next time?
  - Did you need to make any corrections to the code?
- After 15 minutes, share your experience (class discussion)

Style and formatters

Formatter tools

- Writing code in good style is **more than a matter of style**
  - Easier to read→ facilitates collaboration, improves transparency
  - Easier to understand → facilitates debugging and finding mistakes

- There are good style guidelines, esp. the Tidyverse Style Guide
- Re-writing code in good style is time consuming → use **formatters**



```r
rubbish = function(x, y, z) {
  #comment without space
  a_long_fun_call(argument1,
                  c(NA,1,1,1),
                  ...)
                    g(a)%>% b
  new_x <-    y(sqrt(a_logn_fun_call))
  if (x + new_x >3)
  {
    1+11++(
      3/3--3
    )
  }
}
```

An easy-to-use and established tool with great R-Studio integration

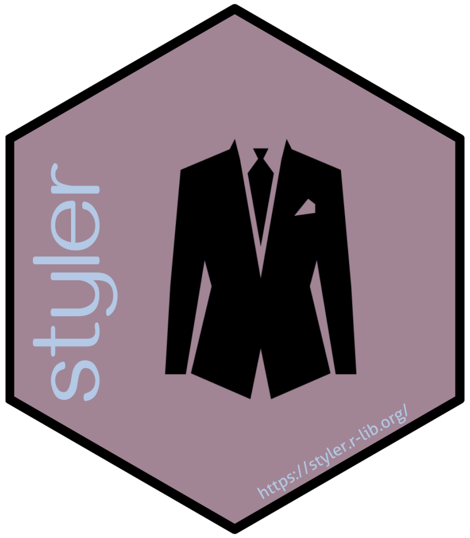
→ Best option for casual users

A modern formatter, extremely fast and powerful, but harder to set up.

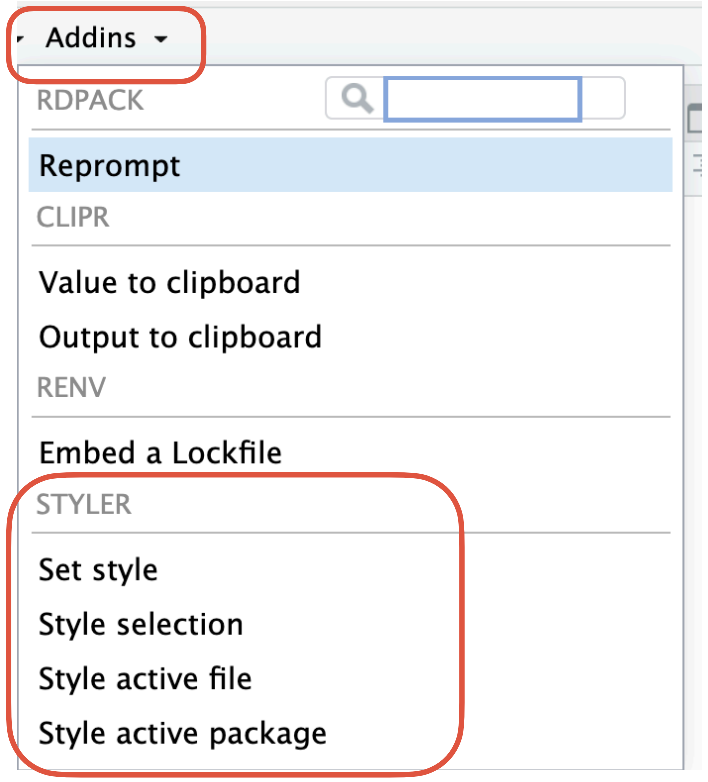
→ Best option for power users

Intermediate task

- Take some code that you have produced so far and style it using `styler`
- Compare the styled and the original - do you like the styled version always better?



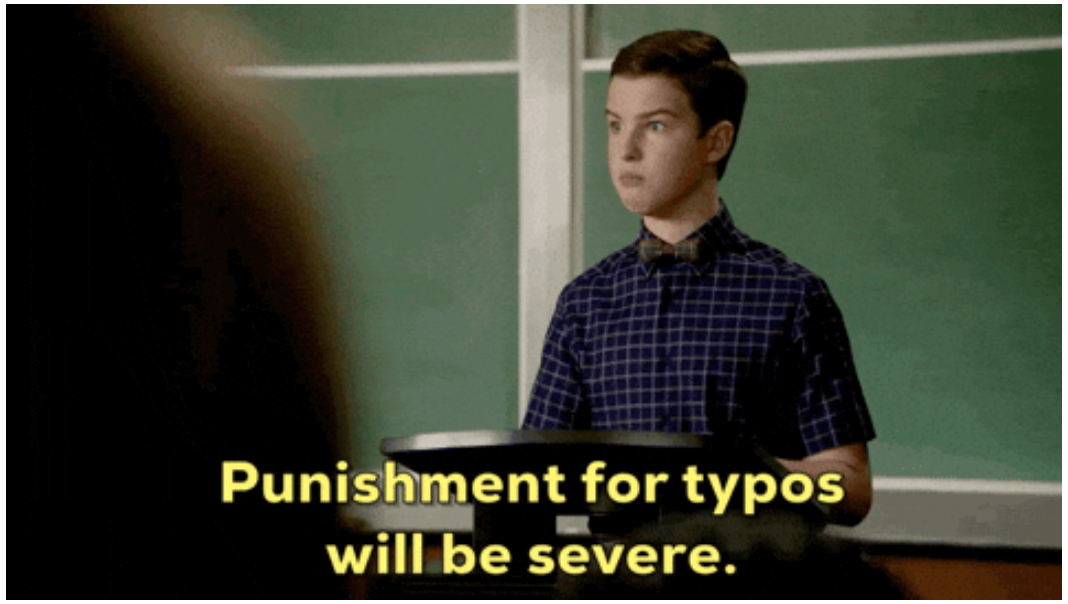
If you haven't installed styler, do:
`install.packages("styler")`
Then it will also be added to the Addins in R-Studio!

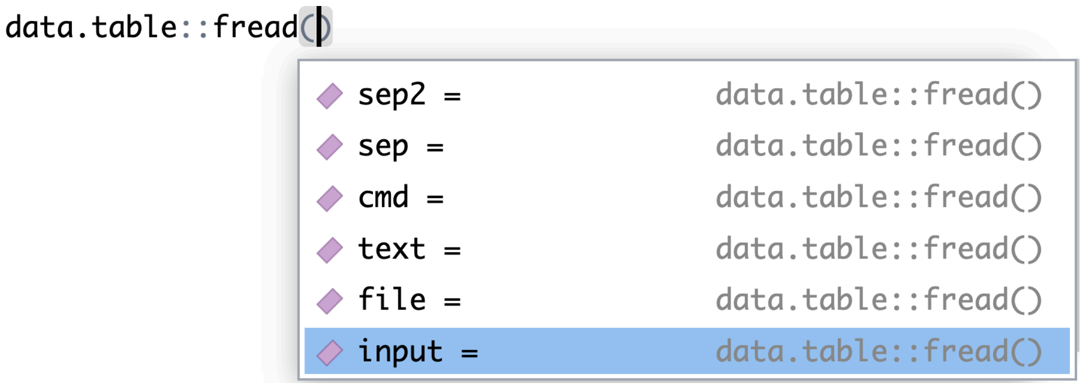


Digression: Avoiding typos

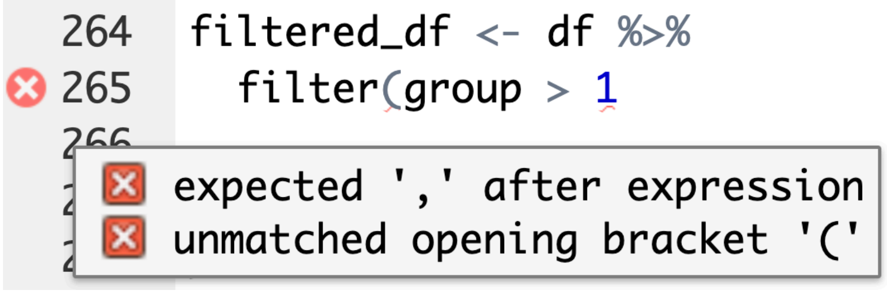
Digression: Tips to avoid typos

- Develop code **step-by-step** and always **test** it regularly
- **Copy-paste** code that has worked
- **Copy-paste** variable names
- Use **auto-completion**
- Make use of the **linting tool** of R-Studio
- **Style tools** such as `styler` also helpful
- If you suspect but cannot find a typo: ask your coding AI of choice!





```
data.table::fread()
  sep2 =     data.table::fread()
  sep =      data.table::fread()
  cmd =      data.table::fread()
  text =     data.table::fread()
  file =     data.table::fread()
  input =    data.table::fread()
```



```
filtered_df <- df %>%
  filter(group > 1
```

```
expected ',' after expression
unmatched opening bracket '('
```

Summary & outlook

Summary and outlook

- Use AI for coding assistance **if you feel comfortable in using the language on your own**
  - Only then AI can boost your productivity enormously
- Many tools exist, often free tiers are sufficient:
  - **Claude**, **ChatGPT** and **Gemini Code Assist** as general tools
  - **Github Copilot** as specific tool with R Studio integration
  - **Windsurf** is often praised, but not specific to R
- You can use AI especially in the areas of **code generation**, **debugging**, **optimisation**, **documentation** and **explanation**
  - Do not expect the final solution, but useful hints and drafts
- There are also non-AI tools that are very useful in practice, e.g., **formatters**

Appendix:

Prompt Engineering for Coding

Prompt Engineering for Coding

The SPEC Framework

	❌	✅
Specify the context	Create a plot of sales data	I have a tibble called `sales_df` with columns: `date` (Date), `revenue` (numeric), `region` (character). Create a `ggplot2` line plot showing revenue over time.
Provide constraints	Calculate average by group	Calculate average alcohol content by wine type using dplyr. Use pipe operators and keep the result in tidyverse format.
Explain the goal	Filter my data	I need to identify outliers in my sales data. Filter for sales values that are more than 2 standard deviations from the mean.
Call for explanation		Explain your code using inline comments for each step. Include an example showing how to use this function.

Prompt Engineering for Coding

Very general blueprint

Help me write R code using [package name].

I have data with these columns:

- [column1]: [type] - [description]
- [column2]: [type] - [description]

I want to [specific goal].

Please use [preferred style, e.g., tidyverse/base R] and add comments explaining each step.

Example

Help me write R code using `dplyr` and `ggplot2`.

I have data with these columns:

- `quarter_date`: Date - quarterly dates from 2020-2024
- `sales`: numeric - sales revenue in dollars
- `region`: character - North, South, East, West

I want to create a faceted line plot showing sales trends for each region with proper labels and a title.

Please use tidyverse pipes and add comments.

Prompt Engineering for Coding

More specific blueprints

Debugging

I want to achieve [X].

I'm getting this error:

[exact error message]

Here's my code:

[paste your code]

Here's my data structure:

[paste str(your_data) or head(your_data)]

Explain what's wrong and how to fix it.

Optimization

I want to achieve [X].

This code works but runs slowly on my dataset of [X rows/size]:

[paste your code]

Please suggest optimizations for speed.

Explain what makes your version faster, and ensure the output remains identical.

Debugging

I found this R code, but don't fully understand it:

[paste code]

Please explain:

1. What does each section do?
2. Why is [specific part] written this way?
3. What would happen if I changed [specific element]?
4. Is it a good implementation?

Prompt Engineering for Coding

Information to include (at least per default)

- **The overall goal:** provide context of the code
- **Data structure:** column names, types, sample values
- **Desired output:** what should the result look like?
- **Package preferences:** base R, tidyverse, data.table?
- **Current errors:** paste exact error messages
- **Your skill level:** "I'm new to ggplot2" or "I'm comfortable with dplyr"

Prompt Engineering for Coding

Iteration is normal

- **Start broad**, get initial code
- **Test it**, identify issues
- **Follow up** with specific problems:
  - This gives a warning about [X], how do I fix it?
  - The output has unexpected NAs in column Y
  - Can you modify this also to handle missing values?
- Don't expect perfection on the first try! Don’t accept unsatisfactory responses!
  - But also: ask critically when it is easier to finalize code yourself

Treat it as a conversation, not a magic spell! 🪄