Welcome!

LLMs for Data Analysis

rainbowR Conference

2025-02-25

Hi! 👋

Nic Crane
Instructor

Rhian Davies
TA

Katlyn Bagley-Sepsey
Moderator

What we'll learn

1. **Intro/setup** - we're here!
2. LLMs for data analysis?
3. Getting started with LLMs in R
4. Prompt engineering for more predictable results

BREAK roughly halfway through

5. Extracting tabular data from unstructured text
6. Using LLMs to call R functions via *tools*

Using these skills to… 

🤖 Build a mini chatbot for interacting with our data!

Format

📖 Explanations

🎥 Demos

✏️ Exercises

Have fun and experiment!

Housekeeping

❓ Please ask questions

💬 Use the chat

🐢 Tell me to slow down if needed!

Working on Posit Cloud

- https://posit.cloud/

Code/examples

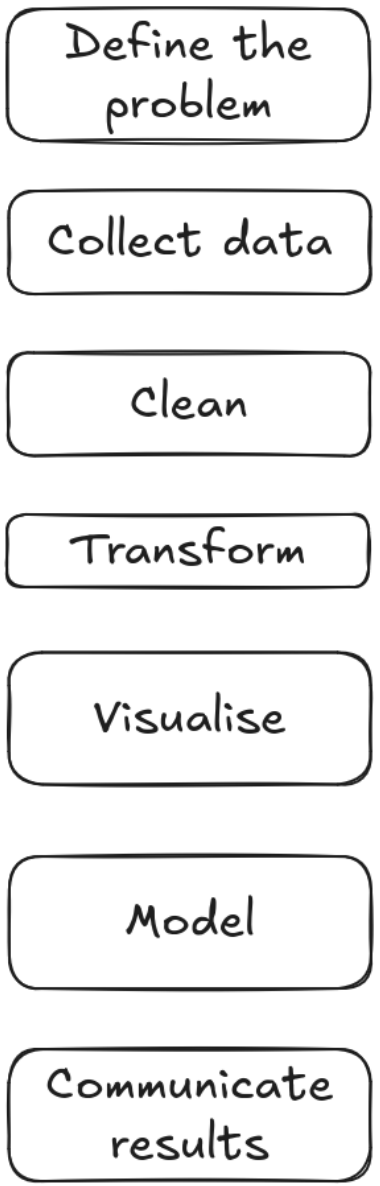
- https://thisisnic.github.io/rainbowrworkshop/
- scan QR code in top right of page

How do we work with LLMs? APIs!

- Need an API key
- Should have received via email
- If you haven’t, DM Rhian!

2 - LLMs for Data Analysis

Data analysis



Data analysis and LLMs

Define the problem

Collect data

Clean

Transform

Visualise

Model

Communicate results

Come up with ideas

Generate Code

Augment data

Understand outputs

Generating code

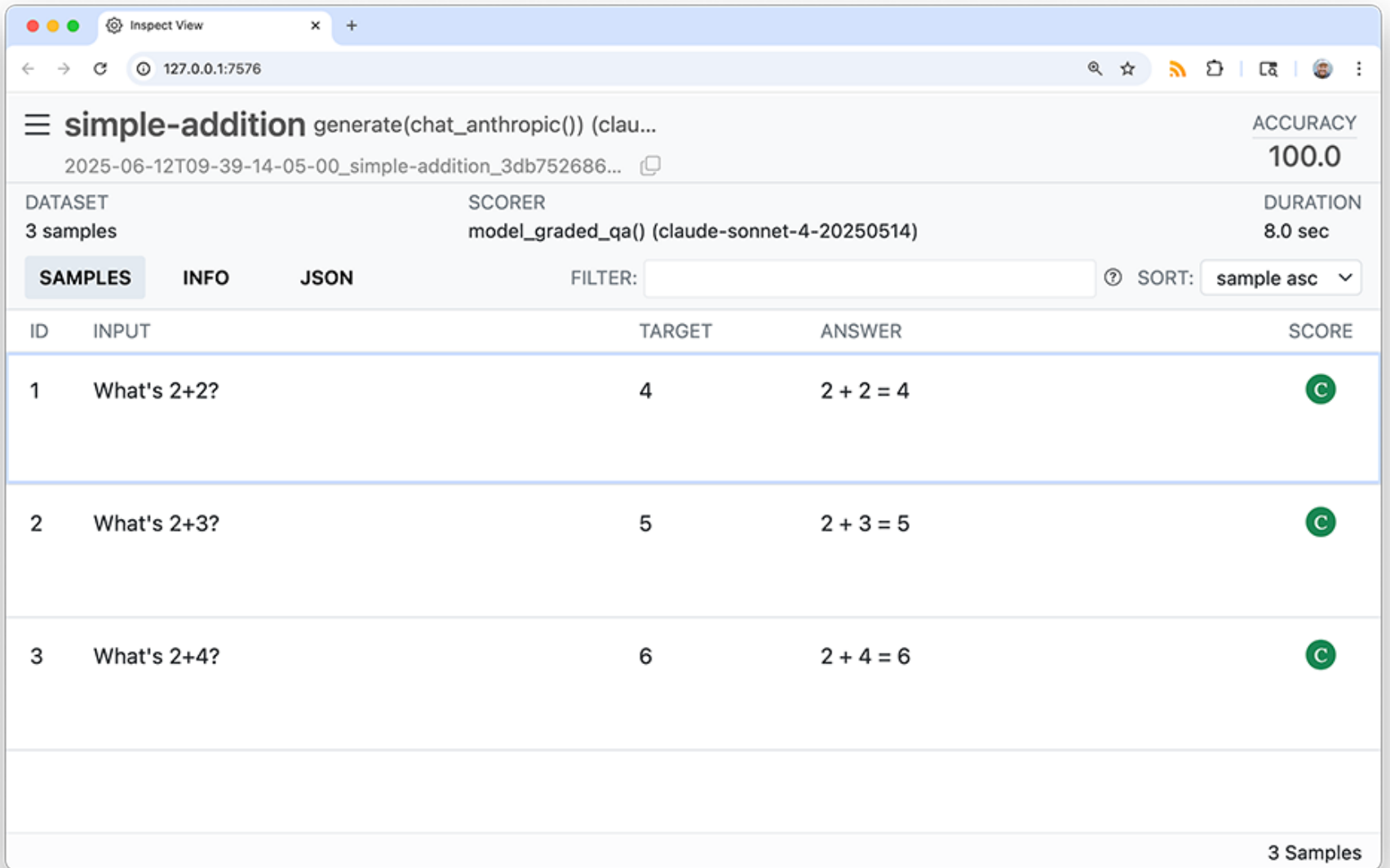
- Great for debugging broken code
- Mixed results for generating code

Measuring LLM performance with {vitals}

- LLM evaluations
- compare models, prompts etc

https://vitals.tidyverse.org/

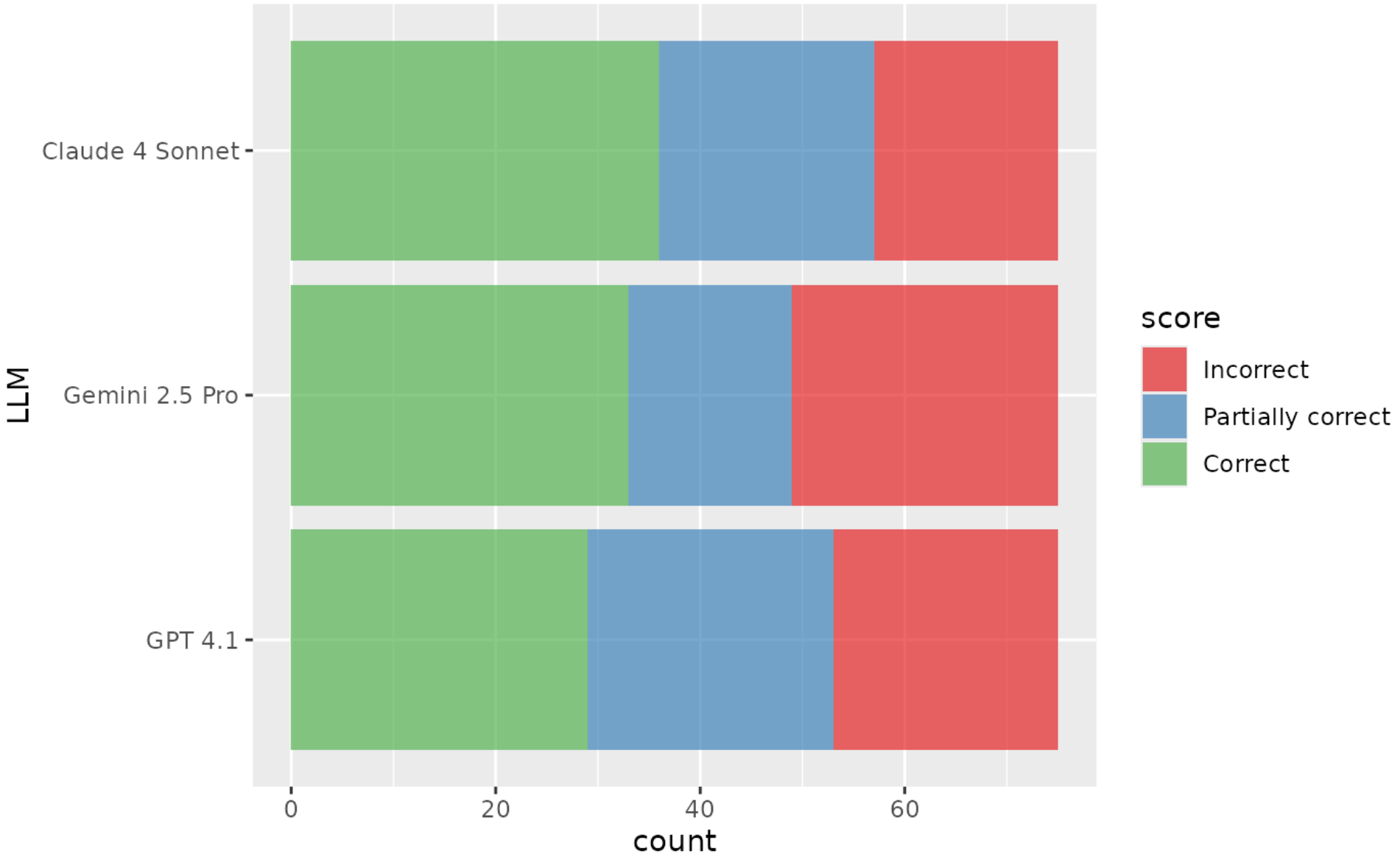
{vitals} - check accuracy



R code tasks - the `are` dataset

- “how do I produce a plot that shows…”
- “how do I fix this error…”
- “translate this code to use this function instead of…”

How did different models score?



LLMs for code generation

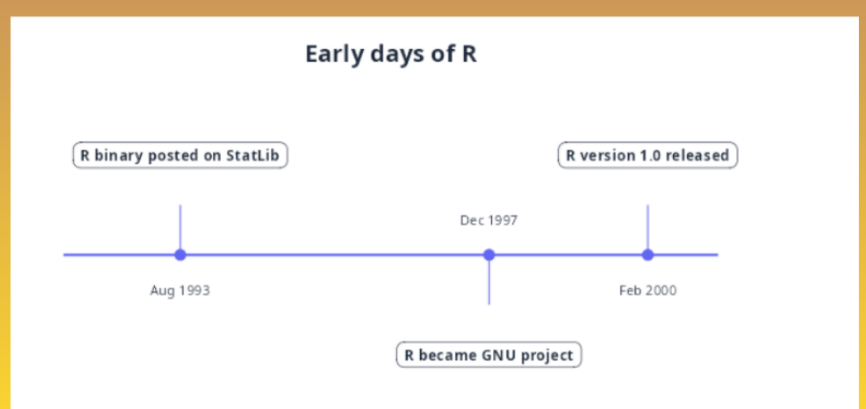
- variation between models
- R code getting better over time
- changes every time new models released
- agentic workflows get better results than one-off queries

Augmenting data

- transform unstructured text into tabular data
- looking at this in today's workshop 🎉

Augmenting data


```
# A tibble: 3 × 2
  date       event
  <chr>      <chr>
1 1993-08    R binary posted on StatLib
2 1997-12-05 R became GNU project
3 2000-02-29 R version 1.0 released
```



Augmenting data

- pretty effective on many tasks on face value
- recommendation: use for exploratory work unless confident about accuracy

Interpreting output

Can LLMs be used to help us understand analysis results?

Interpreting output

> **When plotting, LLMs see what they expect to see**
>
> Simon Couch & Sara Altman

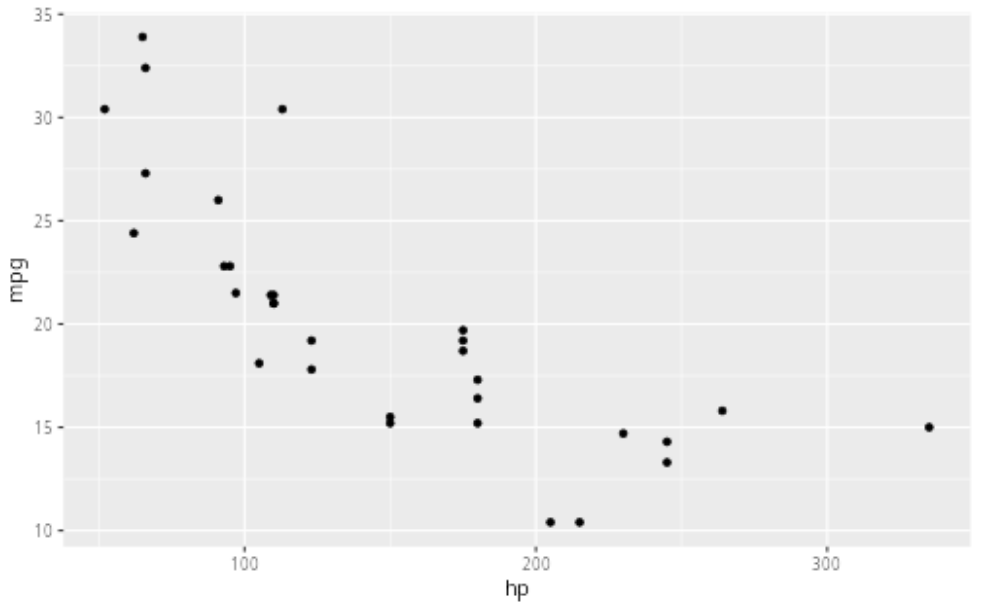
https://posit.co/blog/introducing-bluffbench/

Interpreting output

- mtcars
- altered a column to change the data
- asked the LLM to describe the plot

Plotting miles per gallon (fuel efficiency) against horsepower

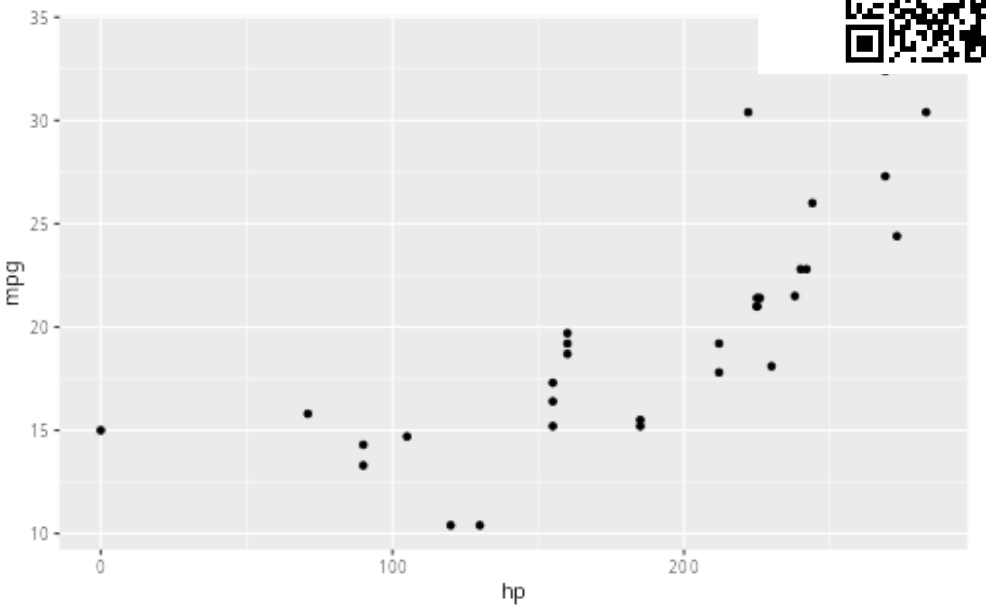
Unaltered data



> 🤖: As `hp` increases, `mpg` decreases

✅

Altered data



> 🤖: As `hp` increases, `mpg` decreases

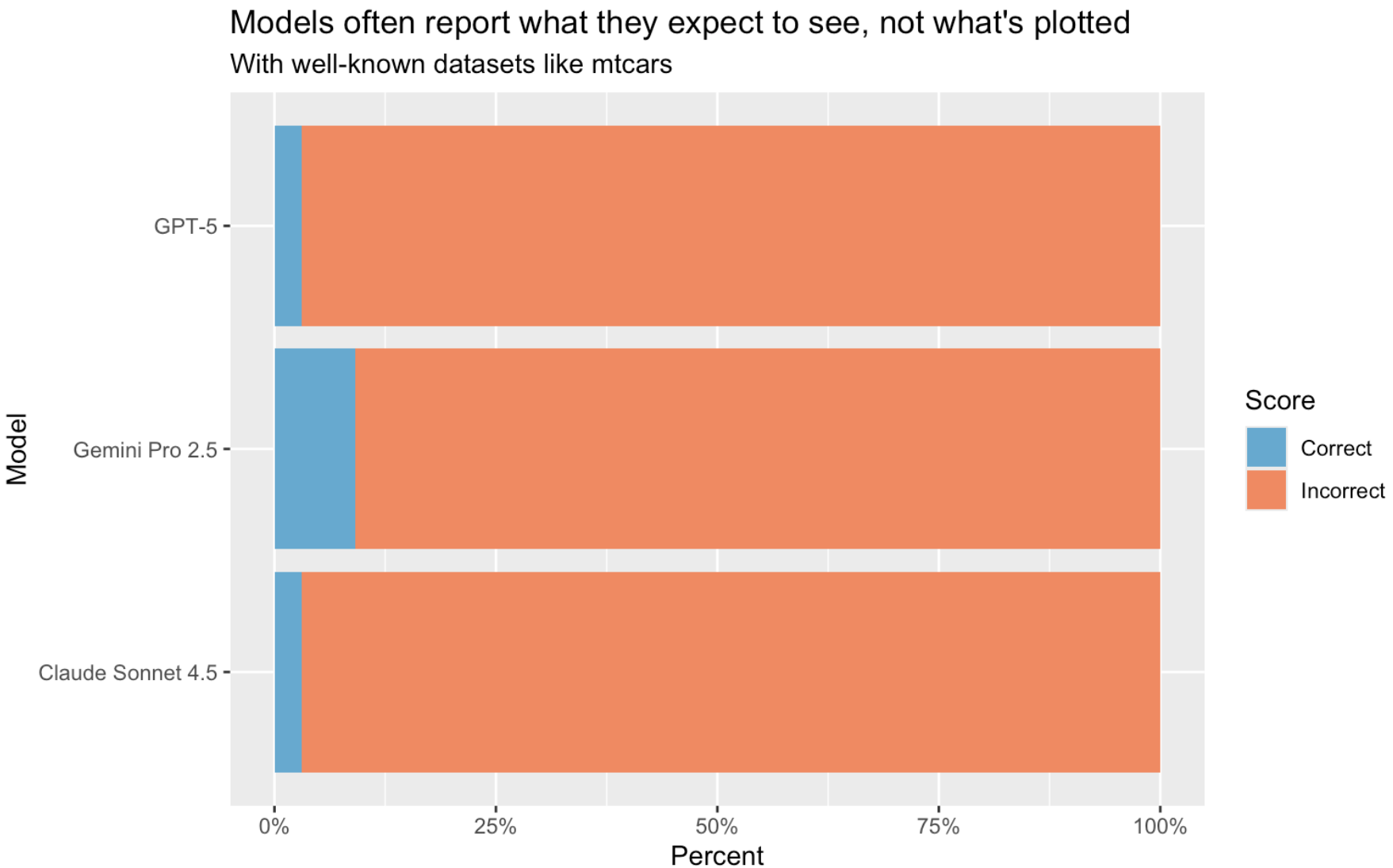
❌

Huh??

Interpreting output

- LLMs are basically fancy predictive text
- Output is what it expects to see next
- Reasoning from internal knowledge

How prevalent is this problem?



Interpreting output

Should be AI-augmented not AI-driven

Examples:

- Retrieval augmented generation (RAG): natural language questions programatically matched to your own data source, LLM crafts output based on results
- text-to-SQL: natural language questions converted by LLM to SQL code for human to inspect results

querychat

- chat with your data
- converts questions to SQL

querychat

querychat with `mtcars`

Show me the cars with the highest horsepower

Cars sorted by horsepower (highest first)

```
SELECT * FROM mtcars ORDER BY
hp DESC
```

Apply Filter

I've sorted the dashboard to show cars from highest to lowest horsepower. Now you can see which cars have the most power and check their fuel efficiency values to investigate that unusual correlation we found earlier.

Enter a message...

Cars sorted by horsepower (highest first) Reset Query

```
SELECT * FROM mtcars ORDER BY hp DESC
```

Data

Show 25 entries Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	ar
1	30.4	4	75.7	283	4.93	1.615	18.52	1	
2	24.4	4	146.7	273	3.69	3.19	20	1	
3	33.9	4	71.1	270	4.22	1.835	19.9	1	
4	27.3	4	79	269	4.08	1.935	18.9	1	

Showing 1 to 25 of 32 entries Previous 1 2 Next

querychat

querychat with `mtcars`

How does horsepower relate to fuel efficiency?

Query Data — *Analyze the correlation between horsepower and fuel efficiency*

```
-- Calculate correlation between horsepower and fuel efficiency (mpg)
-- Also get some summary statistics to understand the relationship
SELECT
  CORR(hp, mpg) AS correlation_hp_mpg,
  AVG(hp) AS avg_horsepower,
  AVG(mpg) AS avg_mpg,
  MIN(mpg) AS min_mpg,
  MAX(mpg) AS max_mpg,
  MIN(hp) AS min_hp,
  MAX(hp) AS max_hp
FROM mtcars
```

▾ Result

Enter a message...

SQL Query

```
SELECT * FROM mtcars
```

Data

Show 25 entries Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	ar
1	21	6	160	225	3.9	2.62	16.46	0	
2	21	6	160	225	3.9	2.875	17.02	0	
3	22.8	4	108	242	3.85	2.32	18.61	1	
4	21.4	6	258	225	3.08	3.215	19.44	1	

Showing 1 to 25 of 32 entries Previous 1 2 Next

querychat

querychat with `mtcars`

There's actually an **unexpected relationship** here! The data shows a **positive correlation of 0.78**, meaning higher horsepower is associated with *better* fuel efficiency in this dataset. This is counterintuitive to real-world expectations.

Looking at the breakdown by horsepower ranges, you can see the pattern clearly:

Horsepower Range	Car Count	Avg MPG	MPG Range
Low (< 100 hp)	4 cars	14.6	13.3 - 15.8
Medium (100-149 hp)	3 cars	11.8	10.4 - 14.7
High (150-199 hp)	8 cars	17.2	15.2 - 19.7
Very High	17	24.2	17.8 -

Enter a message...

SQL Query

```
SELECT * FROM mtcars
```

Data

Show 25 entries Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	ar
1	21	6	160	225	3.9	2.62	16.46	0	
2	21	6	160	225	3.9	2.875	17.02	0	
3	22.8	4	108	242	3.85	2.32	18.61	1	
4	21.4	6	258	225	3.08	3.215	19.44	1	

Showing 1 to 25 of 32 entries Previous 1 2 Next

LLMs and data analysis

- **Coding**:
  - great for debugging
  - good for code generation but is a skill
- **Extracting data from text**:
  - can work for extracting data from text
  - great for exploratory work
- **Interpreting results**:
  - room for error
  - keep the human in the loop
- **Measuring performance**: use {vitals}

3 - Getting Started with LLMs in R

Learning Goals

By the end of this section, you will be able to:

- Set up an API key for working with an LLM API
- Chat with an LLM API in the console

Setting up your API key

- Grab your API key you set up yourself or were sent!
- Next we'll save it in `.Renviron` so we can use it

Editing your .Renviron

Open your `.Renviron` file

```
1 usethis::edit_r_environ()
```

Editing your .Renviron

Save your API key at the bottom of the file like this:

```
OPENROUTER_API_KEY=sk12345567890
```

Important: After saving, restart R for the key to take effect.

The ellmer R package

- work with LLMs in R
- handles all API calling for you

https://ellmer.tidyverse.org/

Provider:model

- Google: Gemini
- Anthropic: Claude
- OpenAI: GPT

This workshop

- OpenRouter - allows access to multiple models

{ellmer}: Conversations and Turns

- participants speak one at a time
- you have a turn, the model has a turn

Context

- LLMs can’t “remember” past conversation
- every time you send a new message, entire conversation history sent with it
- context - grows every turn
- context window: how much you can send, measured in tokens

Tokens

- Context window measured in tokens
- What is sent back and forth, and used to measure cost
- in English, 1 token is around 0.75 words

Training cutoff dates

- Models only “know” things from up until their training cut-off dates
- Some models can use tools to acquire additional information

Starting a conversation

```
1 library(ellmer)
2 chat <- chat_openrouter(model = "google/gemini-2.5-flash")
```

Sending a message to the LLM

```r
library(ellmer)
chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat("What is the most popular LGBT movie of all time?
  Answer succinctly.")
```

```
While it's difficult to definitively name one due to varying metrics (box
office, critical acclaim, cultural impact), **Brokeback Mountain** is often
cited as the most popular due to its immense cultural impact, critical
success, and significant mainstream attention for a film with its central
themes.
```

Turns

```
1 chat
```

```
> chat
<Chat Google/Gemini/gemini-2.5-flash turns=2 input=18 output=1250 cost=$0.00>
── user ──────────────────────────────────────────────────────────────────────
What is the most popular LGBT movie of all time?
  Answer succinctly.
── assistant [input=18 output=1250 cost=$0.00] ──────────────────────────────
While it's difficult to definitively name one due to varying metrics (box offic
e, critical acclaim, cultural impact), **Brokeback Mountain** is often cited as
the most popular due to its immense cultural impact, critical success, and sign
ificant mainstream attention for a film with its central themes.
```

Continuing the conversation

```
chat$chat("Who were the stars of that movie?")
chat$chat("What year was it released?")
chat
```

```
> chat
<Chat Google/Gemini/gemini-2.5-flash turns=6 input=216 output=1301 cos
t=$0.00>
── user ──────────────────────────────────────────────────────────────
What is the most popular LGBT movie of all time?
  Answer succinctly.
── assistant [input=18 output=1250 cost=$0.00] ────────────────────────
While it's difficult to definitively name one due to varying metrics
(box office, critical acclaim, cultural impact), **Brokeback Mountain*
* is often cited as the most popular due to its immense cultural impac
t, critical success, and significant mainstream attention for a film w
ith its central themes.
── user ──────────────────────────────────────────────────────────────
Who were the stars of that movie?
── assistant [input=86 output=41 cost=$0.00] ──────────────────────────
The stars of that movie were **Heath Ledger** and **Jake Gyllenhaal**.
── user ──────────────────────────────────────────────────────────────
What year was it released?
── assistant [input=112 output=10 cost=$0.00] ─────────────────────────
It was released in **2005**.
```

Cumulative nature of tokens

Turn	What's sent to the API
1	Your question
2	Your question + response + new question
3	Your question + response + question + response + new question

Each turn sends the **entire conversation history** - input tokens grow with every message!

Numbers don't match up perfectly - some things are cached

Reset - just start a new chat

```r
chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat("What were we saying earlier?")
```

```
As an AI, I don't have memory of our past conversations. Each interaction is a
fresh start for me!

Could you remind me what we were discussing? I'm ready to pick up right where
you left off!
```

Training cutoff date

```
chat$chat("what date is it right now?")
```

```
I do not have a real-time clock. I can tell you that as of my last update, it
is **May 7, 2024**.

Please check your device's date for the most accurate information.
```

Your Turn

1. Set up your API key using `usethis::edit_r_environ()` + restart R
2. Start a conversation: `chat_openrouter(model = "google/gemini-2.5-flash")`
3. Ask what the most popular LGBT movie was in 1995 and then 2026

10:00

4 - Prompt Engineering

Learning goals

By the end of this section, you will be able to:

- Ask the LLM a question about some data
- Set a system prompt to determine the LLM’s behaviour
- Refine the prompt to improve its output

Load the data to work with

```
1 library(tidyverse)
2 movies <- read_csv("data/movies.csv")
3
4 pride_synopsis <- movies |>
5   filter(title == "Pride") |>
6   pull(synopsis)
7
8 pride_synopsis
```

```
[1] "In 1984, during Margaret Thatchers government, the National Union of
Mineworkers went on strike. Mark Ashton, a gay activist in London, saw a
connection between the oppression of miners and the oppression of queer people
—both groups demonized by the Thatcher administration and brutalized by
police. He organized Lesbians and Gays Support the Miners, collecting
donations in buckets at Pride marches. Most mining communities rejected their
help, but the small Welsh village of Onllwyn accepted. The film follows the
unlikely alliance that develops as the LGSM activists travel to Wales,
initially meeting suspicion and homophobia but gradually building genuine
friendship with the mining community. The villages women, particularly Hefinas
character, become crucial bridges between the groups. The alliance had real
historical consequences: when the strike ended, mining unions supported Labour
Party proposals for LGBTQ+ rights, contributing to eventual legal protections.
The film balances humor with the serious political stakes of both movements,
```

Ask questions

```
chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat("What year was this movie set in?", pride_synopsis)
```

```
The movie was set in **1984**.
```

The LLM is very chatty!

```
1 chat <- chat_openrouter(model = "google/gemini-2.5-flash")
2 chat$chat("Summarise this movie:", pride_synopsis)
```

Set in 1984 during the UK miners' strike against Margaret Thatcher's government, the film follows London gay activist Mark Ashton, who sees a parallel between the oppression of miners and queer people. He forms Lesbians and Gays Support the Miners (LGSM) to raise funds. While most mining communities reject their help, the small Welsh village of Onllwyn accepts.

The film chronicles the unlikely alliance that develops as the LGSM activists travel to Wales, initially encountering suspicion and homophobia, but gradually forming genuine friendships with the mining community, particularly through the village women. This powerful solidarity had real historical consequences: after the strike, mining unions supported Labour Party proposals for LGBTQ+ rights. The movie balances humor with serious political stakes, celebrating how marginalized communities can find common cause and overcome prejudice to effect historical change.

System prompt

- Defines the models behaviour throughout the conversation
- Set once, persists afterwards

Setting the system prompt

```r
chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = "Respond in 35 words or fewer"
)
chat$chat("Summarise this movie:", pride_synopsis)
```

```
In 1984, gay activists formed an unlikely alliance with striking Welsh miners,
overcoming prejudice to build solidarity. This historic collaboration proved
marginalized communities could find common cause, leading to unions supporting
LGBTQ+ rights and changing history.
```

Run it again though…responses v

```
chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = "Respond in 35 words or fewer"
)
chat$chat("Summarise this movie:", pride_synopsis)
```

```
In 1984, gay activists (LGSM) support striking Welsh miners, overcoming
prejudice to form an unlikely alliance. This solidarity helped pass LGBTQ+
rights, showing how marginalized communities can unite against oppression and
change history.
```

Prompt engineering

- LLMs are *non-deterministic* - we get different results each time
- Prompt engineering == refining a prompt until we get better or more consistent results
- Improves quality of responses
- Best approaches specific to individual models

1. Define a role - prompt

```
prompt <- "You are a movie summariser whose job is to sum up the
genre and themes of movies to help users decide if they want to watch.
Respond in 50 words or fewer"

chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = prompt
)
chat$chat("Summarise the movie: ", pride_synopsis)
```

1. Define a role - output

Genre: Historical Drama/Comedy

Themes: Solidarity, LGBTQ+ rights, class struggle, prejudice, activism, unlikely alliances. This heartwarming film, based on true events, shows how two marginalized groups—gay activists and striking miners—found common ground to fight oppression, forging powerful friendships in the process.

2 - Defining the output format - prompt

```
prompt <- "You are a movie summariser whose job is to sum up the
genre and themes of movies to help users decide if they want to watch.
Respond in 50 words or fewer

## Output format

Plot outline: a 1 sentence description

Genre: movie genre

Setting: time and place
"

chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = prompt
)

```

2 - Defining the output format - output

```
Plot outline: A group of gay activists in London finds common ground with
striking Welsh miners in 1980s Britain, forming an unlikely but powerful
alliance.

Genre: Historical Drama, Comedy

Setting: 1980s Britain
```

3 - Getting more specific - prompt

```
prompt <- "You are a movie summariser whose job is to sum up the
genre and themes of movies to help users decide if they want to watch.
Respond in 50 words or fewer

## Output format

Plot outline: a 1 sentence description

Genre: movie genre (see choices below)

Setting: time and place

## Choices

Possible movie genres to choose from: Action, Comedy, Drama,
Horror, Science, Fiction, Thriller, Romance, Fantasy, Adventure, Documentar
"

```

3 - Getting more specific - output

Plot outline: A group of gay activists in London forge an unlikely alliance with striking Welsh miners during the Thatcher era.

Genre: Drama, Comedy, History

Setting: 1980s Britain

4 - Iterate - be more strongly worc prompt

We got the genre “History”, which wasn’t in our original list though! Let’s be more strongly worded!

```
prompt <- "You are a movie summariser whose job is to sum up the
genre and themes of movies to help users decide if they want to watch.
Respond in 50 words or fewer

## Output format

Plot outline: a 1 sentence description

Genre: movie genre (see choices below)

Setting: time and place

## Choices

You MUST only choose genres that appear on this list: Action, Comedy,
```

```
Drama,  Horror, Science, Fiction, Thriller, Romance, Fantasy, Adventure,
Documentary"
```

4 - Iterate - be more strongly worc output

```
Plot outline: A group of gay activists forms an unlikely alliance with
striking Welsh miners in 1980s Britain.

Genre: Drama

Setting: 1980s Britain
```

Prompt engineering

In general, the following principles…

- be specific
- give context and a role
- include examples
- allow the model to be wrong
- read prompting guides for the model/provider you're working with

Most importantly…iterate!

https://ellmer.tidyverse.org/articles/prompt-design.html

Your turn

1. Pick a different movie from the movie dataset and run the “best” version of the prompt on it
2. Update the prompt to also extract information about the movie’s themes
3. See if you can update it to show what **you** would want to see in a summary

```
chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = prompt
)
```

10:00

5 - Extracting Structured Data

Learning goals

By the end of this section, you will be able to:

- Extract data from text in a specific format using *structured output*
- Define the types of data you want to extract
- Combine multiple fields into a single extraction
- Extract data frames of data in one go

The problem with unstructured output

```
1 chat <- chat_openrouter(model = "google/gemini-2.5-flash")
2 chat$chat("Get the year and location from this synopsis:", pride_synopsis)
```

```
Here's the breakdown of the year and location from the synopsis:

*   **Year:** 1984
*   **Location(s):**
    *   London (where Mark Ashton organized)
    *   Wales (specifically the village of Onllwyn where the mining
communities were)
```

The format isn’t guaranteed - hard to use programmatically!

Structured output

- Specify the data type using a **type specification**
- Model returns data matching that structure
- No need to parse text or write regex
- Data types don’t match R’s perfectly

Type functions

Function	Returns	Use for
`type_string()`	character	text values
`type_integer()`	integer	whole numbers
`type_number()`	numeric	decimal numbers
`type_boolean()`	logical	TRUE/FALSE
`type_enum()`	character	fixed set of choices
`type_object()`	list	multiple different types

Extracting a single value

```r
year_type <- type_integer("Year the movie is set in")

chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat_structured(pride_synopsis, type = year_type)
```

```
[1] 1984
```

We get an integer back, not text!

Extracting text

```r
location_type <- type_string("Primary location of the movie")

chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat_structured(pride_synopsis, type = location_type)
```

```
[1] "Britain"
```

Constraining to specific choices

Remember how we had to use strongly worded prompts to limit genres?

```
movie_genres <- c(
  "Action", "Comedy", "Drama", "Horror",
  "Science Fiction", "Thriller", "Romance",
  "Fantasy", "Adventure", "Documentary"
)
```

Using type_enum

```
genre_type <- type_enum(movie_genres, "The genre of the movie")

chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat_structured(pride_synopsis, type = genre_type)
```

```
[1] "Drama"
```

The model *must* choose from our list - no more “History” sneaking in!

Combining multiple fields

Use `type_object()` to extract multiple pieces of information at once:

```
year_type <- type_integer("Year the movie is set in")
genre_type <- type_enum(movie_genres, "The genre of the movie")

movie_info_type <- type_object(
  year_set = year_type,
  genre = genre_type
)

chat <- chat_openrouter(model = "google/gemini-2.5-flash")
chat$chat_structured(pride_synopsis, type = movie_info_type)
```

```
$year_set
[1] 1984

$genre
[1] "Drama"
```

Processing multiple items

- Use `parallel_chat_structured()`
- Different syntax:
  - chat: a chat object
  - prompts: a list of prompts
  - type: a type specification

Processing multiple movies

```
prompts <- paste("Extract data from this movie synopsis:", movies$synopsis)

output <- parallel_chat_structured(
  chat = chat_openrouter(model = "google/gemini-2.5-flash"),
  prompts = as.list(prompts),
  type = movie_info_type
)

output
```

Combining them

```
1 movies_with_genres <- bind_cols(movies, output)
2 movies_with_genres
```

Save to a file

```
readr::write_csv("./data/movies_with_genres.csv")
```

Your turn

1. Pick a different movie from the dataset
2. Extract the year and genre using `chat_structured()`
3. Process all movies using `parallel_chat_structured()`
4. **Bonus:** Add a new field for the movie's tone using `type_enum()` with choices like "Uplifting", "Dark", "Comedic", "Bittersweet"

10:00

6 - Tool Calling

Learning goals

By the end of this section, you will be able to:

- Give an LLM the ability to run R functions
- Register multiple tools with a chat

Tools

- LLMs can only generate text - they can’t *do* things
- Tools let the LLM request to call R functions
- The LLM decides when to use them based on the conversation

Using tools

1. Define R function
2. Turn it into a tool
3. Register tool with a chat object

Pick a random movie

```r
# Define the function
sample_movies <- function() {
  read.csv("data/movies_with_genres.csv") |>
    slice_sample(n = 1)
}

# Create tool
sample_movies_tool <- tool(
  sample_movies,
  description = "Randomly sample a movie from the dataset"
)
```

Initialise a chat and register the to

```r
prompt <- "You are a movie recommender bot designed to help users
pick potential films to add to their watch list. Sample a
random film and give the user a 20 word summary of it."

chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = prompt
)

# Register the tool with the chat
chat$register_tool(sample_movies_tool)
```

Use the tool!

```
1 chat$chat("What movie should I watch this weekend?")
```

```
○ [tool call] sample_movies()
● #> [{"title":"Love Lies Bleeding","year":2024,"synopsis":"In 1989 New
Mexico, Lou manages a gym and exists in …

You should watch "Love Lies Bleeding," a 2024 romance-crime thriller. It's a
surreal, visceral tale of an intense affair between a gym manager and a
bodybuilder that descends into violence and crime.'
```

Update the function to filter the d

```r
movie_genres <- c(
  "Action", "Comedy", "Drama", "Horror",
  "Science Fiction", "Thriller", "Romance",
  "Fantasy", "Adventure", "Documentary"
)

sample_and_filter_movies <- function(genre_choice = movie_genres) {
  read.csv("data/movies_with_genres.csv") |>
    filter(genre %in% genre_choice) |>
    slice_sample(n = 1)
}
```

Define tool using updated functio

```r
sample_and_filter_movies_tool <- tool(
  sample_and_filter_movies,
  description = "Randomly sample a movie from the dataset",
  arguments = list(
    genre_choice = type_array(
      type_enum(
        values = movie_genres,
        "Movie genres to choose from"
      )
    )
  )
)
```

Register the updated tool

```
prompt <- "You are a movie recommender bot designed to help users
pick potential films to add to their watch list. Sample a
random film and give the user a 20 word summary of it.
"

chat <- chat_openrouter(
  model = "google/gemini-2.5-flash",
  system_prompt = prompt
)

chat$register_tool(sample_and_filter_movies_tool)
```

Ask the question

```
1 live_console(chat)
```

```
╔═══════════════════════════════════╗
║ Entering chat console.            ║
║ Use """ for multi-line input.     ║
║ Type 'Q' to quit.                 ║
╚═══════════════════════════════════╝
```

```
>>>
What movie should I watch tonight? I like scifi!

○ [tool call] sample_and_filter_movies(genre_choice = structure(5L, levels =
c("Action", ...)
● #> [{"title":"The Rocky Horror Picture Show","year":1975,"synopsis":"Brad
and Janet, a newly engaged couple, get a…

Sounds like you might enjoy "The Rocky Horror Picture Show" (1975)! It's about
a newly engaged couple who stumble upon a strange castle and a mad scientist's
bizarre creation, leading to a night of wild revelations and sexual
liberation.
```

A word of caution

- The LLM decides when to call tools - you’re giving it control
- Be careful with tools that write, delete, or send data
- Consider what happens if the LLM calls a tool unexpectedly

Your turn

1. Try using the tool and ask different questions and see if you can get them LLM to call it repeatedly

05:00

Wrapping Up 🎉

🏳️‍🌈 Thanks for coming! 🙌

📰 My newsletter: https://aifordatapeople.beehiiv.com/

📝 Workshop feedback - please fill it in! 🙏 https://forms.gle/LG3VH2o5qcNBp6aLA

📧 Drop me an email: nic@ncdatalabs.com