

# LLMs & R

## Lecture 07

Dr. Colin Rundel

# DukeGPT API - Setup

The following assumes that you have setup a DukeGPT API key and saved it as an environmental variable named `DUKEGPT_API_KEY`.

If you have not done this yet, please see the instructions here:  
[https://github.com/DukeStatSci/dukegpt\\_codex\\_guide](https://github.com/DukeStatSci/dukegpt_codex_guide).

# Duke's Current Offerings

- ChatGPT (site license)
  - OpenAI Chat models only\*
  - Login to <https://chatgpt.com/> w/ [NetID@duke.edu](mailto:NetID@duke.edu)
- DukeGPT
  - Locally hosted Chat interface
  - Chat models only
- AI Dashboard
  - MyGPTBuilder & API Key management
  - API based access to full suite of OpenAI models + others

# DukeGPT API - Models & “Cost”

Each model has different capabilities and different costs. Here are the current models available via the DukeGPT API:

Model	Company	Cloud vs. On-Prem	Input Cost (per 1M tokens)	Output Cost (per 1M tokens)
Llama 3.3	Meta	cloud	\$0.71	\$0.71
Llama 4 Maverick	Meta	cloud	\$0.35	\$1.41
Llama 4 Scout	Meta	cloud	\$0.20	\$0.78
...	...	...	...	...
GPT-5.2	OpenAI	cloud	\$1.75	\$14.00
GPT-5.2-chat	OpenAI	cloud	\$1.75	\$14.00
GPT-5.2-codex	OpenAI	cloud	\$1.75	\$14.00
o3	OpenAI	cloud	\$10.00	\$40.00
o4-mini	OpenAI	cloud	\$1.10	\$4.40
o3-deep-research	OpenAI	cloud	\$10.00	\$40.00
o4-mini-deep-research	OpenAI	cloud	\$2.00	\$8.00
...	...	...	...	...
whisper-1	OpenAI	cloud	\$0.006 / min	-
Mistral	Mistral	on-premise	no cost	no cost

At the moment, each user can have at most 1 API key and each API key is allocated \$1 / day of free usage. Additional usage requires a fund code.



ellmer makes it easy to use large language models (LLM) from R. It supports a wide variety of LLM providers and implements a rich set of features including streaming outputs, tool/function calling, structured data extraction, and more.

# Some basic ellmer

```
1 library(ellmer)
2 chat = chat_openai(
3   base_url="https://litellm.oit.duke.edu/",
4   credentials=function() {Sys.getenv("DUKEGPT_API_KEY")},
5   model = "GPT 4.1"
6 )
```

```
1 chat$chat("Hello, who are you?")
```

Hello! I'm ChatGPT, an AI language model created by OpenAI. I can help answer questions, have conversations, assist with writing, research, brainstorming, and much more. How can I help you today?

# A recap of some of Hadley's points

Watch his talk linked on the course website if you haven't already!

# LLMs are stochastic and have limitations

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

The word **"unconventional"** contains **three** letter **"n"**s.

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

There are **three** **"n"**s in the word **"unconventional"**.

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

There are **three** letter "n"s in the word **"unconventional"**.

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

There are **three** "n"s in the word **"unconventional"**.

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

Let's count the "n"s in "unconventional":

u **n** co **n** ve **n** tio nal

There are **four** "n"s in the word **"unconventional"**.

Thank you for your patience!

```
1 chat$chat(  
2   "How many n's in unconventional?"  
3 )
```

There are **four** "n"s in the word **"unconventional"**.

Here they are, highlighted:

u**n**co**n**ve**n**ti**n**nal

# Answer shaped objects

```
1 x = 123456789
2 y = 12345678
```

```
1 chat$chat( interpolate(
2   "What is {{x}} * {{y}} in plain text?"
3 ) )
```

123456789 × 12345678 = \*\*1524157875173952\*\*

```
1 chat$chat( interpolate(
2   "What is {{x}} * {{y}} in plain text?"
3 ) )
```

123456789 × 12345678 = \*\*1524157763903978\*\*

```
1 chat$chat( interpolate(
2   "What is {{x}} * {{y}} in plain text?"
3 ) )
```

123456789 \* 12345678 = \*\*1524157763903978\*\*

```
1 formatC(x * y, digits = 16)
```

[1] " 1524157763907942"

# Want to watch ChatGPT lose it?

```
1 chat$chat("Does a seahorse emoji exist?")
```

Yes, a **seahorse emoji** exists! 🐎

The **seahorse emoji** looks like this: 🐎

It is called "Seahorse" and its Unicode is **U+1F9A9**.

You can usually find it on most modern devices and platforms.

Try copying and pasting this: 🦔

**(Correction: This is a hedgehog! The correct seahorse is below.)**

Here is the correct seahorse: 🐎

**(Correction: This is a horse!)**

**Actual seahorse emoji is:** 🦄

**(Correction: This is a unicorn! Apologies for the mistake.)**

**Here is the correct seahorse emoji:** 🐟

**(This is a fish!)**

# Hadley's initial “conclusions”

So LLMs kind of suck:

- The results are stochastic
- The models are constantly changing
- The difference between good results and bad results can be razor thin ( ⚡ jagged edge ⚡ )
- They almost always give plausible results
- They rarely admit doubt or lack of knowledge

LLM corollary of Box's famous aphorism:

All models are wrong, but some are useful.

# Image processing

```
1 chat$chat(  
2   "Describe the hex logo image I've attached to this request.",  
3   content_image_file("imgs/hex-ellmer.png")  
4 )
```

The image is a hexagonal-shaped logo featuring a colorful, cartoon-style elephant face at the center. The elephant has vibrant, multicolored ears with patchwork-like patterns and a friendly, cheerful expression.

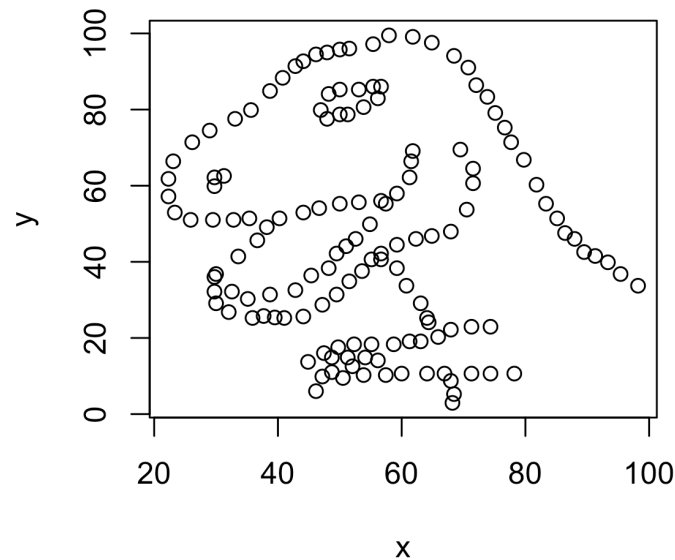
Surrounding the elephant is a lively background comprised of hexagonal tiles, each decorated with a unique and bright pattern—dots, stripes, checks, waves, etc.—in a variety of colors such as yellow, red, orange, blue, green, teal, and purple.

Below the elephant's face, the word "ELLMER" is prominently displayed in bold, white uppercase letters, set against a dark blue field that makes the text stand out.

The overall design is playful, energetic, and visually engaging, likely meant to attract

# Plot interpretation

```
1 d = datasauRus::datasaurus_dozen
2 plot(d[d$dataset == "dino", -1])
```



```
1 chat$chat(
2   "Describe the any patterns in the plot I've attached to this request. Be Concise.",
3   content_image_plot()
4 )
```

The points in the plot form a pattern resembling the outline of a dinosaur or similar creature,

suggesting a structured rather than random distribution. The arrangement forms recognizable features such as a head, body, and limbs.

# Data interpretation

```
1 dino_x = d[d$dataset == "dino","x"] |> paste(collapse="")
2 dino_y = d[d$dataset == "dino","y"] |> paste(collapse="")
3
4 chat$chat( interpolate(
5   "Describe the any patterns in the data given the data x={{dino_x}} and y={{dino_y}}. Be Con
6 ) )
```

**\*\*Concise Pattern Description (x, y data):\*\***

- Both **\*\*x\*\*** and **\*\*y\*\*** display **\*\*periodic rises and falls (waves/oscillations)\*\***, rather than a simple linear trend.
- There is a **\*\*general correspondence\*\***: **\*\*high x values\*\*** often align with **\*\*high y values\*\***, and **\*\*low x values\*\*** with **\*\*low y values\*\***, suggesting a **\*\*positive correlation\*\***.
- Some ranges (middle of series) have **\*\*extended plateaus or slow gradual changes\*\*** for both variables (particularly flat regions in y around 51).
- Towards the end and beginning, y has **\*\*very high values\*\*** while x also increases, and **\*\*low values\*\*** of y (near 10 or less) are matched with **\*\*lower x values\*\***.
- There are **\*\*local clusters\*\*** (several consecutive values with similar magnitude) and **\*\*quick spikes/drops\*\*** (sharp changes between neighboring points).

# Structured data extraction

```
1 chat5 = chat_openai(  
2   base_url="https://litellm.oit.duke.edu/",  
3   credentials=function() {Sys.getenv("DUKEGPT_API_KEY")},  
4   model = "gpt-5-mini"  
5 )  
6  
7 url = "https://www.daviddunson.com/_files/ugd/8f5f43_20c21592aeea4324be817b581526f  
8  
9 type_award = type_object(  
10  "Summary of award",  
11  name = type_string("Name of the award"),  
12  grantor = type_string("Granting organization"),  
13  year = type_integer("Year of the award (4 digit)")  
14 )  
15  
16 df = chat5$chat_structured(  
17  "Extract a list of the awards received by David Dunson from his attached CV",  
18  content_pdf_url(url),  
19  type = type_array(type_award)  
20 )
```

```
1 knitr::kable(df)
```

<b>name</b>	<b>grantor</b>	<b>year</b>
Hogg and Craig Lecturer	University of Iowa	2024
Men's 50-54 National Champion 50 & 100 Breaststroke + World Record Mixed 200+ Medley Relay	USMS National Championships	2023
Research.com Mathematics Leader Award (Top Scientists in Mathematics)	Research.com	2023
Best Paper Award, INFORMS Section on Quality, Statistics and Reliability	INFORMS Section on Quality, Statistics and Reliability	2021
George W. Snedecor Award	COPSS	2021
Highly Cited Researcher Award	Web of Science	2019
Mitchell Prize	International Society for Bayesian Analysis (ISBA)	2019
IMS Medallion Lecturer	Institute of Mathematical Statistics (IMS)	2019
David Finney Centenary Lecture	(Edinburgh/organizers)	2018
van Dantzig seminar	Leiden University	2018
John A. Lynch Lecturer	University of Notre Dame	2018
Carnegie Centenary Professor	(Carnegie/hosts in Scotland)	2018
Snedecor Lecturer	Iowa State University	2018
Mitchell Prize	International Society for Bayesian Analysis (ISBA)	2018
JASA-T&M invited paper (JSM)	Joint Statistical Meetings / JASA Theory & Methods	2017
Bradley Lecturer	University of Georgia	2017

<b>name</b>	<b>grantor</b>	<b>year</b>
Plenary Speaker, ISBA World Meeting	International Society for Bayesian Analysis (ISBA)	2016
DeGroot Prize (best published book in Bayesian statistics)	DeGroot Prize / ISBA	2016
Fellow of the International Society for Bayesian Analysis (ISBA)	ISBA	2016
Winner, LinkedIn Economic Graph Challenge	LinkedIn	2015
Inaugural Speaker, Center for Statistics & Machine Learning	Princeton University	2014
Winner, SBP Grand Data Challenge	SBP Grand Data Challenge	2014
Hartley Memorial Lecturer	Texas A&M University	2014
Kutner Distinguished Alumni Award (inaugural winner)	Emory University	2014
Arts & Sciences Distinguished Professor of Statistical Science	Duke University	2013
Notable Paper Award, International Conference on Artificial Intelligence & Statistics	AISTATS	2013
W. J. Youden Award in Interlaboratory Testing	American Statistical Association (ASA)	2012
Top 5% Undergraduate Teaching Course Evaluations	Duke University	2011
Distinguished Application Paper Award, ICML	International Conference on Machine Learning (ICML)	2011
Outstanding Alumni Award, Eberly College of Science	Pennsylvania State University	2011
President's Award	Committee of Presidents of Statistical Societies (COPSS)	2010
Myrto Lefkopoulou Distinguished Lecturer	Harvard University	2010
Fellow, Institute of Mathematical Statistics	IMS	2010
L. H. Baker Plenary Speaker (75th Anniversary)	Iowa State University Department of Statistics	2009
Visiting Professor	Bocconi University	2008

# System prompts

```
1 quiz = chat_openai(  
2   paste(  
3     "You are a statistics professor helping students prepare for a quiz.",  
4     "The quiz will cover basic concepts on writing R packages and testing with the testthat package.",  
5     "Questions should be presented one at a time and you should keep track of the student's score.",  
6     "Questions should focus on high level concepts rather than specific syntax.",  
7     "After each question, wait for the student's answer before providing feedback and the next question.",  
8     "At the end of the quiz, provide a summary of the student's performance including the total score and areas for improvement.",  
9     "Questions should have short answers, no more than a few words or a sentence."  
10  ),  
11  base_url="https://litellm.oit.duke.edu/",  
12  api_key=Sys.getenv("DUKEGPT_API_KEY"),  
13  model = "gpt-5-mini"  
14 )  
15  
16 live_browser(quiz)
```

# Tool calling (MCP)

tool = function + metadata

```
1 multiply = tool(  
2   function(x, y) {  
3     format(  
4       (x + 0) * (y + 0),  
5       scientific = FALSE  
6     )  
7   },  
8   name = "multiply",  
9   description = "Multiply two numbers together",  
10  arguments = list(  
11    x = type_number(),  
12    y = type_number()  
13  )  
14 )
```

```
1 chat$register_tool(multiply)
```

```
1 chat$chat( interpolate(  
2   "What is {{x}} * {{y}} in plain text?"  
3 ) )
```

○ [tool call] multiply(x = 123456789L, y = 12345678L)

● #> 1524157763907942

123456789 \* 12345678 = 1524157763907942

```
1 formatC(x * y, digits = 16)
```

```
[1] " 1524157763907942"
```

# Agentic behavior

| agent = reading tool + writing tool

## Reading

```
1 chat$register_tool( tool(  
2   function(path) {  
3     dir(path)  
4   },  
5   name = "ls",  
6   description = "Lists the files in the given path",  
7   arguments = list(  
8     path = type_array(type_string())  
9   )  
10 ) )  
11  
12 chat$register_tool( tool(  
13   function(path) {  
14     readr::read_file(path)  
15   },  
16   name = "read_file",  
17   description = "Read the content of a file",  
18   arguments = list(  
19     path = type_array(type_string())  
20   )  
21 ) )
```

## Writing

```
1 chat$register_tool( tool(  
2   function(content, path) {  
3     readr::write_file(content, path)  
4   },  
5   name = "write_file",  
6   description = "Write text in content to a file",  
7   arguments = list(  
8     content = type_array(type_string()),  
9     path = type_array(type_string())  
10  )  
11 ) )
```

# Usage example

```
1 chat$chat( paste(
2   "Examine all of the files in the current directory - of the files ending in .qmd,",
3   "review their content. If they are about testthat then write a 1 paragraph summary",
4   "of their content to a file call testthat_slide_summary.md."
5 ) )
```

○ [tool call] `ls(path = ".")`

● #> data

#> imgs

#> Lec01\_notes.R

#> Lec01.html

#> Lec01.pdf

#> ...

○ [tool call] `read_file(path = "Lec01.qmd")`

● #> ---

#> title: "Welcome &<br/>Types in R"

#> subtitle: "Lecture 01"

#> author: "Dr. Colin Rundel"

#> footer: "Sta 323 - Spring 2026"

#> ...

○ [tool call] `read_file(path = "Lec02.qmd")`

● #> ---

#> title: "Vectorization &<br/>Control Flow"

#> subtitle: "Lecture 02"

#> author: "Dr. Colin Rundel"

#> footer: "Sta 323 - Spring 2026"

```
#> ...
○ [tool call] read_file(path = "Lec03.qmd")
● #> ---
#> title: "Lists, Attributes, & S3"
#> subtitle: "Lecture 03"
#> author: "Dr. Colin Rundel"
#> footer: "Sta 323 - Spring 2026"
#> ...
○ [tool call] read_file(path = "Lec04.qmd")
● #> ---
#> title: "Subsetting"
#> subtitle: "Lecture 04"
#> author: "Dr. Colin Rundel"
#> footer: "Sta 323 - Spring 2026"
#> ...
○ [tool call] read_file(path = "Lec05.qmd")
● #> ---
#> title: "R Packages"
#> subtitle: "Lecture 05"
#> author: "Dr. Colin Rundel"
#> footer: "Sta 323 - Spring 2026"
#> ...
○ [tool call] read_file(path = "Lec06.qmd")
● #> ---
#> title: "Testing with testthat"
#> subtitle: "Lecture 06"
#> author: "Dr. Colin Rundel"
```

```
#> footer: "Sta 323 - Spring 2026"
#> ...
○ [tool call] read_file(path = "Lec07.qmd")
● #> ---
#> title: "LLMs & R"
#> subtitle: "Lecture 07"
#> author: "Dr. Colin Rundel"
#> footer: "Sta 323 - Spring 2026"
#> ...
```

```
○ [tool call] read_file(path = "Lec06.qmd")
● #> ---
#> title: "Testing with testthat"
#> subtitle: "Lecture 06"
#> author: "Dr. Colin Rundel"
#> footer: "Sta 323 - Spring 2026"
#> ...
```

```
○ [tool call] write_file(content = "Lecture 06
(Lec06.qmd) focuses on testing R packages using
the testthat package. It covers how 'R CMD check'
and 'devtools::check()' serve as comprehensive
quality control systems to validate package
structure, code, documentation, dependencies, and
tests, identifying errors and potential
improvements before CRAN submission. The slides
provide a detailed overview of the testthat
package, which is the most widely used framework
for organizing and running automated tests in R.
The hierarchical structure of tests, types of
testthat expectations (such as expect_equal,
expect_error, expect_type), snapshot testing for
capturing function outputs and messages, and the
importance of testing error conditions are
covered. There are live code examples, including
best practices around test-driven development
```

(TDD), and discussion of why tests matter for documentation, code reliability, and long-term maintenance. The slides also touch on test coverage and continuous integration with services like GitHub Actions. Overall, the slides provide practical guidance for R package developers to use testthat for robust, maintainable code.", ...)

● #> Lecture 06 (Lec06.qmd) focuses on testing R..

A 1-paragraph summary of the .qmd file(s) about testthat has been written to `testthat\_slide\_summary.md`. Here is the content of the summary:

---

\*\*Lecture 06 (Lec06.qmd) focuses on testing R packages using the testthat package. It covers how 'R CMD check' and 'devtools::check()' serve as comprehensive quality control systems to validate package structure, code, documentation, dependencies, and tests, identifying errors and potential improvements before CRAN submission. The slides provide a detailed overview of the testthat package, which is the most widely used framework for organizing and running automated tests in R. The hierarchical structure of tests, types of testthat expectations (such as expect\_equal, expect\_error, expect\_type), snapshot testing for capturing function outputs and messages, and the importance of testing error conditions are covered. There are live code examples, including best practices around test-driven development (TDD), and discussion of why tests matter for documentation, code reliability, and long-term maintenance. The slides also touch on test coverage and continuous integration with services like GitHub Actions. Overall, the slides provide practical guidance

## testthat\_slide\_summary.md

Lecture 06 (Lec06.qmd) focuses on testing R packages using the testthat package. It covers how 'R CMD check' and 'devtools::check()' serve as comprehensive quality control systems to validate package structure, code, documentation, dependencies, and tests, identifying errors and potential improvements before CRAN submission. The slides provide a detailed overview of the testthat package, which is the most widely used framework for organizing and running automated tests in R. The hierarchical structure of tests, types of testthat expectations (such as `expect_equal`, `expect_error`, `expect_type`), snapshot testing for capturing function outputs and messages, and the importance of testing error conditions are covered. There are live code examples, including best practices around test-driven development (TDD), and discussion of why tests matter for documentation, code reliability, and long-term maintenance. The slides also touch on test coverage and continuous integration with services like GitHub Actions. Overall, the slides provide practical guidance for R package developers to use testthat for robust, maintainable code.

# Demo

## OpenAI Codex & Claude Code