Package: statlingo (via r-universe)

July 7, 2026

Type Package

Title Explain Statistical Output with Large Language Models

Version 0.1.0

Description Transform complex statistical output into straightforward, understandable, and context-aware natural language descriptions using Large Language Models (LLMs), making complex analyses more accessible to individuals with varying statistical expertise. It relies on the 'ellmer' package to interface with LLM providers including OpenAI <https://openai.com/>, Google AI Studio <https://aistudio.google.com/>, and Anthropic <https://www.anthropic.com/> (API keys are required and managed via 'ellmer').

Depends R (>= 4.1.0)

License GPL (>= 2)

URL https://github.com/bgreenwell/statlingo, https://bgreenwell.github.io/statlingo/

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports ellmer (>= 0.4.1), tools, yaml

Suggests car, gargle, ISLR2, knitr, lme4, lmerTest, MASS, mgcv, nlme, R6, rmarkdown, survival, tibble, tinytest

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/pak/sysreqs libssl-dev

Repository https://bgreenwell.r-universe.dev

Date/Publication 2026-07-07 02:28:47 UTC

RemoteUrl https://github.com/bgreenwell/statlingo

RemoteRef main

RemoteSha 9debd8e6d2e293a2901c20cabe380344a5411b6d

RemoteSubdir r

Contents

`explain` *Explain statistical output*

Description

Use an LLM to explain the output from various statistical objects using straightforward, understandable, and context-aware natural language descriptions.

Usage

```
explain(
  object,
  client,
  context = NULL,
  audience = c("novice", "student", "researcher", "manager", "domain_expert"),
  verbosity = c("moderate", "brief", "detailed"),
  style = c("markdown", "html", "json", "text", "latex"),
  language = NULL,
  prompt_dir = NULL,
  ...
)

## Default S3 method:
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)
```

```
## S3 method for class 'htest'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'lm'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'glm'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'polr'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
```

```
  prompt_dir = NULL,
  ...
)

## S3 method for class 'lme'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'lmerMod'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'glmerMod'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'gam'
explain(
  object,
  client,
  context = NULL,
```

```
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'survreg'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'coxph'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)

## S3 method for class 'rpart'
explain(
  object,
  client,
  context = NULL,
  audience = "novice",
  verbosity = "moderate",
  style = "markdown",
  language = NULL,
  prompt_dir = NULL,
  ...
)
```

Arguments

`object`	An appropriate statistical object. For example, `object` can be the output from calling t.test() or glm().
`client`	A Chat object (e.g., from calling chat_openai() or [chat_gemini()][ellmer::chat_gemini)]). [ellmer::chat_gemini)]: R:ellmer::chat_gemini)
`context`	Optional character string providing additional context, such as background on the research question and information about the data.
`audience`	Character string indicating the target audience: • `"novice"` - Assumes the user has a limited statistics background (default). • `"student"` - Assumes the user is learning statistics. • `"researcher"` - Assumes the user has a strong statistical background and is familiar with common methodologies. • `"manager"` - Assumes the user needs high-level insights for decision-making. • `"domain_expert"` - Assumes the user is an expert in their own field but not necessarily in statistics.
`verbosity`	Character string indicating the desired verbosity: • `"moderate"` - Offers a balanced explanation (default). • `"brief"` - Offers a high-level summary. • `"detailed"` - Offers a comprehensive interpretation.
`style`	Character string indicating the desired output style: • `"markdown"` (default) - Output formatted as plain Markdown. • `"html"` - Output formatted as an HTML fragment. • `"json"` - Output structured as a JSON string parseable into an R list. • `"text"` - Output as plain text. • `"latex"` - Output as a LaTeX fragment.
`language`	Character string specifying the language the explanation should be written in (e.g. `"Spanish"`, `"French"`, `"Mandarin Chinese"`). If `NULL` (the default), no language constraint is added and the LLM will typically respond in the same language as the input/context or its default language.
`prompt_dir`	Optional character string specifying a custom directory containing custom prompt templates to override or overlay the package's default templates.
`...`	Additional optional arguments. (Currently ignored.)

Details

The following models and package classes are supported:

- **stats** (Base R):
  - `htest` (Hypothesis tests, e.g., t.test(), wilcox.test(), cor.test())
  - `lm` (Linear regression models via lm())
  - `glm` (Generalized linear models via glm())
- **MASS**:

  - `polr` (Proportional odds logistic regression via polr())
- **nlme**:
  - `lme` (Linear mixed-effects models via lme())
- **lme4**:
  - `lmerMod` (Linear mixed-effects models via lmer())
  - `glmerMod` (Generalized linear mixed-effects models via glmer())
- **mgcv**:
  - `gam` (Generalized additive models via gam())
- **survival**:
  - `survreg` (Parametric survival regression via survreg())
  - `coxph` (Cox proportional hazards models via coxph())
- **rpart**:
  - `rpart` (Recursive partitioning decision trees via rpart())

Value

An object of class `"statlingo_explanation"`. Essentially a list with the following components:

- `text` - Character string representation of the LLM's response.
- `model_type` - Character string giving the internal prompt model type (e.g., `"linear_model"` or `"cox_proportional_hazards"`).
- `audience` - Character string specifying the level or intended audience for the explanations.
- `verbosity` - Character string specifying the level of verbosity or level of detail of the provided explanation.

Examples

```
## Not run:
# Polynomial regression
fm1 <- lm(dist ~ poly(speed, degree = 2), data = cars)
context <- "
The data give the speed of cars (mph) and the distances taken to stop (ft).
Note that the data were recorded in the 1920s!
"
# Use Google Gemini to explain the output; requires an API key; see
# ?ellmer::chat_google_gemini for details
client <- ellmer::chat_google_gemini(echo = "none")
ex <- explain(fm1, client = client, context = context)
explain(fm1, client = client, context = context, language = "Spanish")

# Poisson regression example using the bike sharing data from ISLR2
Bikeshare <- ISLR2::Bikeshare

# Fit a Poisson regression model to the bike sharing data set
fm2 <- glm(bikers ~ mnth + hr + workingday + temp + weathersit,
           data = Bikeshare, family = poisson)
```

```
# Additional context for the LLM to consider when explaining the model's
# output
context <- "
The data contain the hourly and daily count of rental bikes between years
2011 and 2012 in Capital bikeshare system, along with weather and seasonal
information. The variables in the model include:

* bikers - Total number of bikers.
* mnth - Month of the year, coded as a factor.
* hr - Hour of the day, coded as a factor from 0 to 23.
* workingday - Is it a work day? Yes=1, No=0.
* temp - Normalized temperature in Celsius. The values are derived via
  (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39.
* weathersit - Weather, coded as a factor.
"

# Use Google Gemini to explain the output; requires an API key; see
# ?ellmer::chat_google_gemini for details
client <- ellmer::chat_google_gemini(echo = "none")
explain(fm2, client = client, context = context, audience = "student",
        verbosity = "brief", style = "text")

## End(Not run)
```

`print.statlingo_explanation` *Print LLM explanation*

Description

Print a formatted version of an LLMs explanation using `cat()`.

Usage

```
## S3 method for class 'statlingo_explanation'
print(x, ...)
```

Arguments

`x`	A statlingo_explanation object.
`...`	Additional optional arguments to be passed to `print.default()`.

Value

Invisibly returns the printed statlingo_explanation object.

suggest_code *Suggest next statistical coding steps*

Description

Suggest code snippets to run next based on a model explanation.

Usage

```
suggest_code(x, ...)
```

Arguments

x	A statlingo_explanation object (returned by explain()).
...	Additional arguments.

Value

An object of class statlingo_code_suggestions.

Examples

```
## Not run:
fm <- lm(dist ~ speed, data = cars)
client <- ellmer::chat_google_gemini()
ex <- explain(fm, client = client)
suggest_code(ex)

## End(Not run)
```

summarize *Summarize statistical output*

Description

Generate text-based summaries of statistical output that can be embedded into prompts for querying Large Language Models (LLMs). Intended primarily for internal use.

Usage

```
summarize(object, ...)

## Default S3 method:
summarize(object, ...)

## S3 method for class 'htest'
```

```
summarize(object, ...)

## S3 method for class 'lm'
summarize(object, ...)

## S3 method for class 'glm'
summarize(object, ...)

## S3 method for class 'polr'
summarize(object, ...)

## S3 method for class 'lme'
summarize(object, ...)

## S3 method for class 'lmerMod'
summarize(object, ...)

## S3 method for class 'glmerMod'
summarize(object, ...)

## S3 method for class 'gam'
summarize(object, ...)

## S3 method for class 'survreg'
summarize(object, ...)

## S3 method for class 'coxph'
summarize(object, ...)

## S3 method for class 'rpart'
summarize(object, ...)
```

Arguments

`object`	An object for which a summary is desired (e.g., a glm object).
`...`	Additional optional arguments. (Currently ignored.)

Value

A character string summarizing the statistical output.

See Also

summary().

Examples

```
tt <- t.test(1:10, y = c(7:20))
summarize(tt)  # prints output as a character string
```

```
cat(summarize(tt))  # more useful for reading
```

Index