

Package: EnTraineR (via r-universe)

May 17, 2026

Type Package

Title Enhanced Teaching Assistant (AI) for Statistical Analysis

Version 1.0.0

Description An assistant built on large language models that helps interpret statistical model outputs in R by generating concise, audience-specific explanations.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ollamar, httr2 (>= 1.0.0)

Suggests FactoMineR, commonmark, rmarkdown

SystemRequirements Pandoc (for DOCX/HTML conversion when using rmarkdown)

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

URL <https://github.com/Sebastien-Le/EnTraineR>

BugReports <https://github.com/Sebastien-Le/EnTraineR/issues>

Config/pak/sysreqs libssl-dev

Repository <https://sebastien-le.r-universe.dev>

Date/Publication 2026-01-12 08:07:33 UTC

RemoteUrl <https://github.com/sebastien-le/entrainer>

RemoteRef HEAD

RemoteSha e011c9b0d336c5508f408cdb70750e038db55720

Contents

deforestation	2
gemini_generate	3
ham	4

poussin	6
trainer_AovSum	6
trainer_chisq_test	8
trainer_cor_test	9
trainer_core_actually_shown	10
trainer_core_audience_profile	11
trainer_core_build_prompt	12
trainer_core_conf_label	13
trainer_core_detect_main_factors	13
trainer_core_extract_block_after	14
trainer_core_filter_ttest_by_factors	14
trainer_core_generate_or_return	15
trainer_core_llm_generate	16
trainer_core_prompt_header	16
trainer_core_summary_only_block	17
trainer_core_ttest_scope_msg	17
trainer_LinearModel	18
trainer_MCA	19
trainer_PCA	21
trainer_prop_test	22
trainer_t_test	23
trainer_var_test	24

Index	26
--------------	-----------

deforestation	<i>River deforestation: air and water temperatures before/after</i>
---------------	---

Description

Monitoring data of water and air temperatures before and after riparian deforestation. Useful to illustrate linear regression with an interaction (Temp_air * Deforestation).

Usage

```
data(deforestation)
```

Format

A data frame with 56 rows and 3 variables:

Temp_water numeric; water temperature (deg C).

Temp_air numeric; air temperature (deg C).

Deforestation factor with 2 levels: "BEFORE", "AFTER". 28 periods each.

Details

Brief summary (indicative): Temp_water min ~ 0.55, median ~ 9.28, max ~ 18.89; Temp_air min ~ -3.04, median ~ 6.53, max ~ 15.75.

Examples

```
data(deforestation)
str(deforestation)
table(deforestation$Deforestation)

# Linear model with interaction (FactoMineR):
fit <- FactoMineR::LinearModel(
  Temp_water ~ Temp_air * Deforestation,
  data = deforestation,
  selection = "none"
)
print(fit)
```

gemini_generate	<i>Generate text with Google Gemini (Generative Language API) - robust w/ retries</i>
-----------------	---

Description

Minimal wrapper around the Generative Language API 'generateContent' endpoint for text prompts, with retries, exponential backoff, clearer errors, and optional output compilation (HTML/DOCX) with auto-open.

Usage

```
gemini_generate(
  prompt,
  model = "gemini-2.5-flash",
  api_key = Sys.getenv("GEMINI_API_KEY"),
  user_agent = NULL,
  base_url = "https://generativelanguage.googleapis.com/v1beta",
  temperature = NULL,
  top_p = NULL,
  top_k = NULL,
  max_output_tokens = NULL,
  stop_sequences = NULL,
  system_instruction = NULL,
  safety_settings = NULL,
  seed = NULL,
  timeout = 120,
  verbose = FALSE,
  max_tries = 5,
  backoff_base = 0.8,
  backoff_cap = 8,
  force_markdown = TRUE,
  compile_to = c("none", "html", "docx")
)
```

Arguments

prompt	Character scalar. The user prompt (plain text).
model	Character scalar. Gemini model id (e.g., "gemini-2.5-flash", "gemini-2.5-pro"). You may also pass "models/..." and it will be normalized.
api_key	Character scalar. API key. Defaults to env var 'GEMINI_API_KEY'.
user_agent	Character scalar. If NULL, a dynamic value is used.
base_url	Character scalar. API base URL.
temperature	Optional numeric in [0, 2].
top_p	Optional numeric in (0, 1].
top_k	Optional integer ≥ 1 .
max_output_tokens	Optional integer > 0 .
stop_sequences	Optional character vector.
system_instruction	Optional character scalar.
safety_settings	Optional list passed as-is to the API.
seed	Optional integer seed.
timeout	Numeric seconds for request timeout (default 120).
verbose	Logical; if TRUE, prints URL/retries.
max_tries	Integer. Max attempts (default 5).
backoff_base	Numeric. Initial backoff seconds (default 0.8).
backoff_cap	Numeric. Max backoff seconds (default 8).
force_markdown	Logical. If TRUE, instructs the model to answer in Markdown.
compile_to	Character scalar. One of c("none", "html", "docx").

Value

If `compile_to = "none"`: character scalar (raw text as returned by the API). If `compile_to = "html"`: `list(markdown = <string>, html_path = <path>)`, and opens the HTML. If `compile_to = "docx"`: `list(markdown = <string>, docx_path = <path>)`, and opens the DOCX.

 ham

Ham: sensory descriptors and overall liking

Description

Sensory profile of hams (quantitative attributes) and an overall liking score. Useful to illustrate multiple regression and the joint reading of per-term F tests and coefficient T tests.

Usage

```
data(ham)
```

Format

A data frame with 21 rows (hams) and 15 variables:

Juiciness numeric

Crispy numeric

Tenderness numeric

Pasty numeric

Fibrous numeric

Salty numeric

Sweet numeric

Meaty numeric

Seasoned numeric

Metallic numeric

Ammoniated numeric

Fatty numeric

Braised numeric

Lactic numeric

Overall liking numeric; overall acceptability score

Details

Brief summary (indicative): median Juiciness ~ 3.0; median Tenderness ~ 6.0; mean Salty ~ 5.52; median Overall liking ~ 6.5.

Examples

```
data(ham)
summary(ham)
```

```
# Multiple regression without selection (FactoMineR):
fit <- FactoMineR::LinearModel(
  `Overall liking` ~ .,
  data = ham,
  selection = "none"
)
print(fit)
```

poussin

Poussin: weight by brooding temperature and sex

Description

Chick weights measured under three brooding temperatures, with sex recorded. Useful for ANOVA and linear models with categorical factors.

Usage

```
data(poussin)
```

Format

A data frame with 45 rows and 3 variables:

Temperature factor with 3 levels: "T1", "T2", "T3" (15 each).

Gender factor with 2 levels: "Female", "Male" (about 20 and 25).

Weight numeric; weight (units as provided).

Details

Brief summary (indicative): Weight min ~ 15, median ~ 23, max ~ 33.

Examples

```
data(poussin)
with(poussin, table(Temperature, Gender))
boxplot(Weight ~ Temperature, data = poussin,
        main = "Poussin weight by temperature")
# Two-factor ANOVA (base stats):
fit <- stats::aov(Weight ~ Temperature * Gender, data = poussin)
summary(fit)
```

trainer_AovSum*Trainer: Interpret ANOVA (AovSum) with an LLM-ready prompt*

Description

Builds an English-only, audience-tailored prompt to interpret an ANOVA produced by FactoMineR::AovSum. The function never invents numbers: it only passes verbatim excerpts to the LLM and instructs how to interpret deviations (sum-to-zero coding) as performance drivers.

Usage

```

trainer_AovSum(
  aovsum_obj,
  introduction = NULL,
  alpha = 0.05,
  t_test = NULL,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

aovsum_obj	An object whose printed output contains sections named "Ftest" and "Ttest" (e.g., FactoMineR::AovSum()).
introduction	Optional character context paragraph for the analysis. Defaults to a generic description.
alpha	Numeric significance level used as an instruction for the LLM. Default 0.05.
t_test	Optional character vector to filter the T-test section by factor names and/or interactions (e.g. "Factor A" or "Factor A:Factor B").
audience	Target audience, one of c("beginner", "applied", "advanced").
summary_only	Logical; if TRUE, return a compact 3-bullet executive summary.
llm_model	Character model name for the generator (e.g., "llama3").
generate	Logical; if TRUE, calls trainer_core_generate_or_return().

Value

Character prompt (if generate = FALSE) or a list.

Examples

```

## Not run:
# Example 1: SensoMineR chocolates (requires SensoMineR)
if (requireNamespace("SensoMineR", quietly = TRUE)) {
  # Load data from SensoMineR
  data("chocolates", package = "SensoMineR")
  # ANOVA summary with Product and Panelist
  res <- FactoMineR::AovSum(Granular ~ Product * Panelist, data = sensochoc)

  intro <- "Six chocolates have been evaluated by a sensory panel,
  during two days, according to a sensory attribute: granular.
  The panel has been trained according to this attribute
  and panellists should be reproducibile when rating this attribute."
  intro <- gsub("\n", " ", intro)
  intro <- gsub("\\s+", " ", intro)
  cat(intro)
}

```

```

prompt <- trainer_AovSum(res, audience = "beginner",
                        t_test = c("Product", "Panelist"),
                        introduction = intro)
cat(prompt)

res <- gemini_generate(prompt, compile_to = "html")
}

# Example 2: Poussin dataset (shipped with this package)
data(poussin)
intro <- "For incubation, 45 chicken eggs were randomly assigned to three batches of 15.
Three treatments (different incubation temperatures) were then applied to the batches.
We assume that after hatching, all chicks were raised under identical conditions
and then weighed at a standard reference age.
At that time, the sex of the chicks - a factor known beforehand to cause
significant weight differences - could also be observed.
The objective is to choose the treatment that maximizes chick weight."
intro <- gsub("\n", " ", intro)
intro <- gsub("\s+", " ", intro)
cat(intro)

res <- FactoMineR::AovSum(Weight ~ Gender * Temperature, data = poussin)

prompt <- trainer_AovSum(res,
                        audience = "beginner",
                        t_test = c("Gender", "Temperature"),
                        introduction = intro)
cat(prompt)

res <- gemini_generate(prompt, compile_to = "html")

## End(Not run)

```

trainer_chisq_test *Interpret a chi-squared test (chisq.test) with an audience-aware LLM prompt*

Description

Builds a clear, audience-tailored prompt to interpret base R stats::chisq.test() results, handling both goodness-of-fit and contingency-table tests. Aligned with other TraineR trainers: no invented numbers; audience-specific guidance.

Usage

```

trainer_chisq_test(
  csq_obj,
  introduction = NULL,
  alpha = 0.05,

```

```

  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

csq_obj	An htest object returned by stats::chisq.test().
introduction	Optional character string giving the study context.
alpha	Numeric significance level (default 0.05).
audience	One of c("beginner", "applied", "advanced").
summary_only	Logical; if TRUE, return a 3-bullet executive summary regardless of audience depth (uses trainer_core_summary_only_block()).
llm_model	Character; model name for the generator (default "llama3").
generate	Logical; if TRUE, call the generator and return prompt + response.

Value

If generate = FALSE, a prompt string. If TRUE, a list with prompt, response, and model.

Examples

```

# GOF
set.seed(1); x <- c(18, 22, 20, 25, 15)
csq1 <- chisq.test(x, p = rep(1/5, 5))
cat(trainer_chisq_test(csq1, audience = "beginner"))

# Contingency (independence)
tbl <- matrix(c(12,5,7,9), nrow=2)
csq2 <- chisq.test(tbl) # Yates for 2x2 by default
cat(trainer_chisq_test(csq2, audience = "applied"))

```

trainer_cor_test	<i>Interpret a correlation test (cor.test) with an audience-aware LLM prompt</i>
------------------	--

Description

Builds a clear, audience-tailored prompt to interpret stats::cor.test() results for Pearson, Spearman, or Kendall correlation. Supports three audiences ("beginner", "applied", "advanced") and an optional summary_only mode.

Usage

```

trainer_cor_test(
  ct_obj,
  introduction = NULL,
  alpha = 0.05,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

ct_obj	An htest object returned by stats::cor.test().
introduction	Optional character string giving the study context.
alpha	Numeric significance level (default 0.05).
audience	One of c("beginner","applied","advanced").
summary_only	Logical; if TRUE, return a 3-bullet executive summary regardless of audience depth (uses trainer_core_summary_only_block()).
llm_model	Character; model name for the generator (default "llama3").
generate	Logical; if TRUE, call the generator and return prompt + response.

Value

If generate = FALSE, a prompt string. If TRUE, a list with prompt, response, and model.

Examples

```

set.seed(1)
x <- rnorm(30); y <- 0.5*x + rnorm(30, sd = 0.8)
ct <- cor.test(x, y, method = "pearson")
cat(trainer_cor_test(ct, audience = "applied", summary_only = FALSE))

```

trainer_core_actually_shown

Determine which requested items were actually shown after filtering

Description

Determine which requested items were actually shown after filtering

Usage

```

trainer_core_actually_shown(req_main, req_inter, ttest_filtered)

```

Arguments

req_main Character vector of requested main factors.
 req_inter Character vector of requested interactions ("A:B").
 ttest_filtered Filtered T-test lines.

Value

Character vector of actually shown specifiers.

Examples

```
trainer_core_actually_shown("A", "A:B", c("A - a", "A - a : B - b"))
```

```
trainer_core_audience_profile
      Build an audience profile (beginner / applied / advanced) with optional
      summary-only mode
```

Description

Build an audience profile (beginner / applied / advanced) with optional summary-only mode

Usage

```
trainer_core_audience_profile(
  audience = c("beginner", "applied", "advanced"),
  alpha = 0.05,
  summary_only = FALSE
)
```

Arguments

audience Character: one of c("beginner","applied","advanced").
 alpha Numeric alpha (only to instruct the LLM; no computation).
 summary_only Logical; if TRUE, enforce a short 3-bullet executive summary regardless of audience depth.

Value

List with flags, tone, and guardrails: - audience, summary_only, tone - show_verbatim, show_diagnostics - include_df, include_equations - max_bullets, max_words_takeaway - guard, alpha_round

Examples

```
trainer_core_audience_profile("applied", 0.05, summary_only = FALSE)
```

`trainer_core_build_prompt`*Assemble a standard prompt with common sections*

Description

Assemble a standard prompt with common sections

Usage

```
trainer_core_build_prompt(  
    header,  
    context,  
    setup,  
    verbatim,  
    output_requirements,  
    show_verbatim = TRUE,  
    verbatim_title = "Verbatim output"  
)
```

Arguments

<code>header</code>	Character (from <code>trainer_core_prompt_header</code>).
<code>context</code>	Character (short paragraph).
<code>setup</code>	Character (bullet list or short lines).
<code>verbatim</code>	Character (raw printed output, will be fenced).
<code>output_requirements</code>	Character (audience-tailored instructions).
<code>show_verbatim</code>	Logical; include verbatim block.
<code>verbatim_title</code>	Character section title.

Value

Full prompt string.

Examples

```
trainer_core_build_prompt("H", "Context", "- a\n- b", "raw", "Reqs")
```

trainer_core_conf_label
Confidence level label helper

Description

Returns a short label for a confidence level, e.g. "95". If conf_level is NA or NULL, returns fallback.

Usage

```
trainer_core_conf_label(conf_level, fallback = "the reported")
```

Arguments

conf_level Numeric in (0,1), or NA/NULL.
fallback Character string to use when conf_level is missing. Default is "the reported".

Value

A character scalar such as "95%" or the fallback string.

Examples

```
trainer_core_conf_label(0.95)  
trainer_core_conf_label(NA)  
trainer_core_conf_label(NULL, fallback = "not reported")
```

trainer_core_detect_main_factors
Detect main-effect factor names present in T-test lines (ignore interactions) Space-safe: captures everything before " - " on non-interaction rows.

Description

Detect main-effect factor names present in T-test lines (ignore interactions) Space-safe: captures everything before " - " on non-interaction rows.

Usage

```
trainer_core_detect_main_factors(tt_lines)
```

Arguments

tt_lines Character vector.

Value

Character vector of factor names.

trainer_core_extract_block_after

Extract lines following a header (up to first blank line)

Description

Extract lines following a header (up to first blank line)

Usage

```
trainer_core_extract_block_after(txt, header)
```

Arguments

txt	Printed object as a single string.
header	Exact header text to search.

Value

Character vector of lines until first blank line.

Examples

```
trainer_core_extract_block_after("Head\nA\n\nB", "Head")
```

trainer_core_filter_ttest_by_factors

Filter T-test lines by requested factors (main and/or interactions)

Description

Filter T-test lines by requested factors (main and/or interactions)

Usage

```
trainer_core_filter_ttest_by_factors(  
  tt_lines,  
  keep_factors = NULL,  
  keep_intercept = TRUE  
)
```

Arguments

tt_lines Character vector of T-test lines.
keep_factors Character vector of factor names or "A:B".
keep_intercept Logical; keep (Intercept) line.

Value

Filtered character vector.

Examples

```
trainer_core_filter_ttest_by_factors(c("(Intercept)", "A - a", "A - b:B - c"), "A", TRUE)
```

```
trainer_core_generate_or_return
```

Generate or return a prompt, depending on 'generate'

Description

Generate or return a prompt, depending on 'generate'

Usage

```
trainer_core_generate_or_return(prompt, llm_model = "llama3", generate = FALSE)
```

Arguments

prompt Character prompt to return or send.
llm_model Character model name.
generate Logical flag.

Value

Character prompt or list(prompt, response, model).

Examples

```
trainer_core_generate_or_return("hello", "llama3", generate = FALSE)
```

```
trainer_core_llm_generate
```

LLM generation helper for TraineR

Description

Thin wrapper around the chosen LLM backend. By default, uses **ollamar** if installed; otherwise returns only the prompt so the caller can still inspect it without failing.

Usage

```
trainer_core_llm_generate(model, prompt, engine = c("ollamar", "none"), ...)
```

Arguments

model	Character scalar, model name (e.g., "llama3").
prompt	Character scalar, the prompt to send.
engine	Character scalar, backend engine. Currently "ollamar" or "none". If "none" or if the backend is not available, returns the prompt only.
...	Passed to the backend generator.

Value

A list with elements prompt, response, model, and engine. If the backend isn't available, response is NULL.

```
trainer_core_prompt_header
```

Build the standard header for prompts

Description

Build the standard header for prompts

Usage

```
trainer_core_prompt_header(profile)
```

Arguments

profile	List from trainer_core_audience_profile().
---------	--

Value

Character header.

Examples

```
pr <- trainer_core_audience_profile("applied", 0.05)
cat(trainer_core_prompt_header(pr))
```

```
trainer_core_summary_only_block
```

Utility: render a standard 3-bullet summary-only instruction

Description

Utility: render a standard 3-bullet summary-only instruction

Usage

```
trainer_core_summary_only_block(
  words_limit = 50,
  bullets = 3,
  label = "the analysis"
)
```

Arguments

words_limit	Integer maximum total words (default 50).
bullets	Integer number of bullets (default 3).
label	Character label to include (e.g., the test name).

Value

Character instruction block.

Examples

```
cat(trainer_core_summary_only_block(50, 3, "t-test"))
```

```
trainer_core_ttest_scope_msg
```

Scope message for T-test section based on requested & found factors

Description

Scope message for T-test section based on requested & found factors

Usage

```
trainer_core_ttest_scope_msg(t_test, requested, actually_shown)
```

Arguments

t_test User request vector.
 requested Vector of normalized requested items.
 actually_shown Vector from trainer_core_actually_shown().

Value

Single character scope message.

Examples

```
trainer_core_ttest_scope_msg(c("A"), c("A"), c("A"))
```

```
trainer_LinearModel    Trainer: Interpret FactoMineR::LinearModel with an LLM-ready prompt
```

Description

Builds an English-only, audience-tailored prompt to interpret a FactoMineR::LinearModel result. Handles model selection (AIC/BIC) and instructs how to interpret deviation contrasts (sum-to-zero) for factors. Works for ANOVA, ANCOVA, and multiple regression.

Usage

```
trainer_LinearModel(  
  lm_obj,  
  introduction = NULL,  
  alpha = 0.05,  
  t_test = NULL,  
  audience = c("beginner", "applied", "advanced"),  
  summary_only = FALSE,  
  llm_model = "llama3",  
  generate = FALSE  
)
```

Arguments

lm_obj An object returned by FactoMineR::LinearModel(...).
 introduction Optional character string giving the study context.
 alpha Numeric significance level (default 0.05).
 t_test Optional character vector to filter the T-test section by factor names and/or interactions (e.g. "FactorA" or "FactorA:FactorB").
 audience One of c("beginner", "applied", "advanced").
 summary_only Logical; if TRUE, return a 3-bullet executive summary.
 llm_model Character model name for the generator (e.g., "llama3").
 generate Logical; if TRUE, call the generator.

Value

Character prompt or list.

Examples

```
# --- Example 1: multiple regression with selection (ham) -----
data(ham)
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  intro_ham <- "A sensory analysis institute wants to know if it's possible to predict
the overall liking of a ham from its sensory description.
A trained panel used the following attributes to describe 21 hams:
Juiciness, Crispy, Tenderness, Pasty, Fibrous, Salty, Sweet, Meaty,
Seasoned, Metallic, Ammoniated, Fatty, Braised, Lactic.
Afterward, an Overall Liking score was assigned to each of the hams."
# collapse whitespace safely without extra packages
intro_ham <- gsub("\n", " ", intro_ham)
intro_ham <- gsub("\\s+", " ", intro_ham)

res <- FactoMineR::LinearModel(`Overall liking` ~ ., data = ham, selection = "bic")
pr <- trainer_LinearModel(res, introduction = intro_ham, audience = "advanced",
                          generate = FALSE)

cat(pr)
}

# --- Example 2: interaction with a categorical factor (deforestation) -----
data(deforestation)
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  intro_flume <- "The study's goal is to determine how river deforestation affects
the relationship between water and air temperature.
The dataset contains maximum air and water temperatures measured over
28 ten-day periods before deforestation and 28 periods after deforestation.
The main objective is to understand if and how the link between air and
water temperature changes after deforestation."
intro_flume <- gsub("\n", " ", intro_flume)
intro_flume <- gsub("\\s+", " ", intro_flume)

res <- FactoMineR::LinearModel(Temp_water ~ Temp_air * Deforestation,
                              data = deforestation, selection = "none")
pr <- trainer_LinearModel(res, introduction = intro_flume, audience = "advanced",
                          generate = FALSE)

cat(pr)
}
```

Description

Builds an English-only, audience-tailored prompt to name and justify a Multiple Correspondence Analysis (MCA) dimension from a FactoMineR::MCA object. The function never invents numbers: it passes verbatim excerpts from `summary(mca_obj)` and `FactoMineR::dimdesc()` filtered at a given significance threshold `proba`, and instructs how to read and name the axis.

Usage

```
trainer_MCA(
  mca_obj,
  dimension = 1L,
  proba = 0.05,
  introduction = NULL,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)
```

Arguments

<code>mca_obj</code>	A MCA object returned by <code>FactoMineR::MCA()</code> .
<code>dimension</code>	Integer scalar; the dimension (component) to name (default 1).
<code>proba</code>	Numeric in (0,1]; significance threshold used by <code>FactoMineR::dimdesc()</code> to characterize the dimension (default 0.05).
<code>introduction</code>	Optional character string giving the study context. Defaults to a generic description.
<code>audience</code>	One of <code>c("beginner", "applied", "advanced")</code> .
<code>summary_only</code>	Logical; if TRUE, return a compact 3-bullet executive summary (uses <code>trainer_core_summary_only_block</code>).
<code>llm_model</code>	Character; model name for your generator backend (default "llama3").
<code>generate</code>	Logical; if TRUE, calls <code>trainer_core_generate_or_return()</code> and returns a list with prompt, response, and model. If FALSE, returns the prompt string.

Value

If `generate = FALSE`, a character prompt string. If `generate = TRUE`, a list with prompt, response, and model.

Examples

```
## Not run:
# Example: tea (FactoMineR)
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  data(tea, package = "FactoMineR")
  res_mca <- FactoMineR::MCA(tea, quanti.sup = 19, quali.sup = 20:36, graph = FALSE)

  intro <- "A survey on tea consumption practices and contexts was summarized by MCA."
```

```

intro <- gsub("\n", " ", intro); intro <- gsub("\s+", " ", intro)

# Applied audience
prompt <- trainer_MCA(res_mca,
                      dimension = 1,
                      proba = 0.01,
                      introduction = intro,
                      audience = "applied",
                      generate = FALSE)

cat(prompt)

res <- gemini_generate(prompt, compile_to = "html")
}

## End(Not run)

```

trainer_PCA	<i>Trainer: Name a PCA dimension (FactoMineR::PCA) with an LLM-ready prompt</i>
-------------	---

Description

Builds an English-only, audience-tailored prompt to name and justify a principal component (dimension) from a FactoMineR::PCA object. The function never invents numbers: it passes verbatim excerpts from ‘summary(pca_obj)’ (Individuals/Variables) and ‘FactoMineR::dimdesc()’ filtered at a given significance threshold ‘proba’, and instructs how to read and name the axis.

Usage

```

trainer_PCA(
  pca_obj,
  dimension = 1L,
  proba = 0.05,
  introduction = NULL,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

pca_obj	A PCA object returned by FactoMineR::PCA().
dimension	Integer scalar; the dimension (component) to name (default 1).
proba	Numeric in (0,1]; significance threshold used by FactoMineR::dimdesc() to characterize the dimension (default 0.05).
introduction	Optional character string giving the study context. Defaults to a generic description.

audience	One of c("beginner", "applied", "advanced").
summary_only	Logical; if TRUE, return a compact 3-bullet executive summary (uses trainer_core_summary_only_block).
llm_model	Character; model name for your generator backend (default "llama3").
generate	Logical; if TRUE, calls trainer_core_generate_or_return() and returns a list with prompt, response, and model. If FALSE, returns the prompt string.

Value

If generate = FALSE, a character prompt string. If generate = TRUE, a list with prompt, response, and model.

Examples

```
## Not run:
# Example: decathlon (FactoMineR)
if (requireNamespace("FactoMineR", quietly = TRUE)) {
  data(decathlon, package = "FactoMineR")

  res_pca <- FactoMineR::PCA(decathlon,
    quanti.sup = 11:12,
    quali.sup = 13,
    graph = FALSE)

  intro <- "A study was conducted on decathlon athletes.
  Performances on each event were measured and summarized by PCA."
  intro <- gsub("\n", " ", intro); intro <- gsub("\s+", " ", intro)

  prompt <- trainer_PCA(res_pca,
    dimension = 1,
    proba = 0.05,
    introduction = intro,
    audience = "applied",
    generate = FALSE)

  cat(prompt)

  res <- gemini_generate(prompt, compile_to = "html")
}

## End(Not run)
```

trainer_prop_test	<i>Interpret a proportion test (prop.test) with an audience-aware LLM prompt</i>
-------------------	--

Description

Builds a clear, audience-tailored prompt to interpret stats::prop.test() results (one-sample vs target p, two-sample equality, k-group equality, or k-group vs given p). Aligned with other TraineR trainers: no invented numbers; audience-specific guidance.

Usage

```

trainer_prop_test(
  pt_obj,
  introduction = NULL,
  alpha = 0.05,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

pt_obj	An htest object returned by stats::prop.test().
introduction	Optional character string giving the study context.
alpha	Numeric significance level (default 0.05).
audience	One of c("beginner","applied","advanced").
summary_only	Logical; if TRUE, return a 3-bullet executive summary regardless of audience depth (uses trainer_core_summary_only_block()).
llm_model	Character; model name for the generator (default "llama3").
generate	Logical; if TRUE, call the generator and return prompt + response.

Value

If generate = FALSE, a prompt string. If TRUE, a list with prompt, response, and model.

Examples

```

# One-sample
pt1 <- prop.test(x = 56, n = 100, p = 0.5)
cat(trainer_prop_test(pt1, audience = "beginner"))

# Two-sample
pt2 <- prop.test(x = c(42, 35), n = c(100, 90))
cat(trainer_prop_test(pt2, audience = "applied", summary_only = TRUE))

```

 trainer_t_test

Interpret a Student's t-test (stats::t.test) with an LLM-ready prompt

Description

Builds a clear, audience-tailored prompt to interpret a base R stats::t.test() result. Identifies the test flavor (One-sample, Two-sample, Paired, Welch) and instructs the LLM to use ONLY printed values (p, t, df, CI, estimates) and avoid any new calculations.

Usage

```

trainer_t_test(
  tt_obj,
  introduction = NULL,
  alpha = 0.05,
  audience = c("beginner", "applied", "advanced"),
  summary_only = FALSE,
  llm_model = "llama3",
  generate = FALSE
)

```

Arguments

tt_obj	An htest object returned by <code>stats::t.test()</code> .
introduction	Optional character string giving the study context in plain English.
alpha	Numeric significance level used for interpretation (default 0.05).
audience	One of <code>c("beginner", "applied", "advanced")</code> .
summary_only	Logical; if TRUE, return a 3-bullet executive summary.
llm_model	Character; model name passed to your generator (default "llama3").
generate	Logical; if TRUE, call <code>trainer_core_generate_or_return()</code> and return prompt + response.

Value

If `generate = FALSE`, the prompt string. Else a list with prompt, response, model.

Examples

```

set.seed(1)
tt1 <- t.test(rnorm(20, 0.1), mu = 0)           # one-sample
cat(trainer_t_test(tt1, audience = "beginner"))

x <- rnorm(18, 0); y <- rnorm(20, 0.3)
tt2 <- t.test(x, y, var.equal = FALSE)        # two-sample Welch
cat(trainer_t_test(tt2, audience = "applied", summary_only = TRUE))

```

trainer_var_test	<i>Interpret an F test comparing two variances (var.test) with an audience-aware LLM prompt</i>
------------------	---

Description

Builds a clear, audience-tailored prompt to interpret a base R `stats::var.test()` result.

Usage

```
trainer_var_test(  
  vt_obj,  
  introduction = NULL,  
  alpha = 0.05,  
  audience = c("beginner", "applied", "advanced"),  
  summary_only = FALSE,  
  llm_model = "llama3",  
  generate = FALSE  
)
```

Arguments

vt_obj	An htest object returned by stats::var.test().
introduction	Optional character string giving the study context.
alpha	Numeric significance level (default 0.05).
audience	One of c("beginner", "applied", "advanced").
summary_only	Logical; if TRUE, return a 3-bullet executive summary regardless of audience depth (uses trainer_core_summary_only_block()).
llm_model	Character; model name for the generator (default "llama3").
generate	Logical; if TRUE, call the generator and return prompt + response.

Value

If generate = FALSE, a prompt string. If TRUE, a list with prompt, response, and model.

Examples

```
set.seed(1)  
x <- rnorm(25, sd = 1.0); y <- rnorm(30, sd = 1.3)  
vt <- var.test(x, y)  
cat(trainer_var_test(vt, audience = "applied"))  
cat(trainer_var_test(vt, audience = "advanced", summary_only = TRUE))
```

Index

* datasets

deforestation, [2](#)

ham, [4](#)

poussin, [6](#)

deforestation, [2](#)

gemini_generate, [3](#)

ham, [4](#)

poussin, [6](#)

trainer_AovSum, [6](#)

trainer_chisq_test, [8](#)

trainer_cor_test, [9](#)

trainer_core_actually_shown, [10](#)

trainer_core_audience_profile, [11](#)

trainer_core_build_prompt, [12](#)

trainer_core_conf_label, [13](#)

trainer_core_detect_main_factors, [13](#)

trainer_core_extract_block_after, [14](#)

trainer_core_filter_ttest_by_factors,
[14](#)

trainer_core_generate_or_return, [15](#)

trainer_core_llm_generate, [16](#)

trainer_core_prompt_header, [16](#)

trainer_core_summary_only_block, [17](#)

trainer_core_ttest_scope_msg, [17](#)

trainer_LinearModel, [18](#)

trainer_MCA, [19](#)

trainer_PCA, [21](#)

trainer_prop_test, [22](#)

trainer_t_test, [23](#)

trainer_var_test, [24](#)