Package: AIGENIE (via r-universe)

May 24, 2026

Type Package

Title Automatic Item Generation and Validation via Network-Integrated Evaluation

Version 2.1.0

Date 2026-03-10

Description Automated psychological scale development and structural validation using large language models (LLMs) and network psychometric methods. Implements the AI-GENIE framework (Automatic Item Generation and Validation via Network-Integrated Evaluation) to generate candidate items, compute embedding representations, and estimate dimensional structure using Exploratory Graph Analysis (EGA). Item quality is evaluated using Unique Variable Analysis to identify redundant items and Bootstrap EGA to assess item and dimension stability. Supports both fully automated item generation and analysis of user-provided item sets, facilitating efficient, theory-informed measurement development prior to empirical data collection.

License AGPL (>= 3)

Encoding UTF-8

Depends R (>= 3.6.0), EGAnet (>= 2.4.0)

Imports reticulate, ggplot2, igraph, patchwork, jsonlite

Suggests testthat (>= 3.0.0), tictoc

URL https://github.com/laralee/AIGENIE

BugReports https://github.com/laralee/AIGENIE/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3

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

Date/Publication 2026-05-22 18:29:38 UTC

RemoteUrl https://github.com/laralee/AIGENIE

RemoteRef HEAD

RemoteSha e72bb46f2965d2dcfe6aa7a859b6b8c1c56257a7

Contents

`AIGENIE` *Generate, Validate, and Check Items using AI-GENIE*

Description

Generate, validate, and check your items for quality and redundancy using AI-GENIE (Generative Psychometrics via AI-GENIE: Automatic Item Generation and Validation via Network-Integrated Evaluation). AI-GENIE is a methodology that combines the latest open-source LLMs and generative artificial intelligence with advances in network psychometrics to facilitate scale generation, selection, and validation. The pipeline eliminates the need to generate hundreds of items by content experts, recruit diverse and experienced researchers, administer items to thousands of participants, and employ modern psychometric methods in the collected data.

Usage

```
AIGENIE(
  item.attributes,
  openai.API = NULL,
  hf.token = NULL,
  main.prompts = NULL,
  groq.API = NULL,
  anthropic.API = NULL,
  jina.API = NULL,
  model = "gpt4o",
```

```
  temperature = 1,
  top.p = 1,
  embedding.model = "text-embedding-3-small",
  target.N = NULL,
  domain = NULL,
  scale.title = NULL,
  item.examples = NULL,
  audience = NULL,
  item.type.definitions = NULL,
  response.options = NULL,
  prompt.notes = NULL,
  system.role = NULL,
  EGA.model = NULL,
  EGA.algorithm = NULL,
  EGA.uni.method = NULL,
  uva.cut.off = 0.2,
  keep.org = FALSE,
  items.only = FALSE,
  embeddings.only = FALSE,
  adaptive = TRUE,
  run.overall = FALSE,
  all.together = FALSE,
  plot = TRUE,
  silently = FALSE
)
```

Arguments

`item.attributes`
: A named list of atomic character vectors (required). Describes the attributes or characteristics that each item type should encompass. These are not necessarily lower-order dimensions, but can be if the item types represent appropriate hierarchical constructs. Each nested list must have at least two unique attributes. Repeated attributes within the same nested list are not allowed, but attributes can be repeated across nested lists. Each name of the list must be unique, and all elements within sublists must be strings. For example, attributes of the personality trait "neuroticism" might include "anxious", "depressed", "insecure", or "emotional" since these characteristics encompass aspects of neuroticism that the item pool should address.

`openai.API`
: A character string or NULL (optional, default: NULL). The OpenAI API key for authentication with OpenAI's services. Required when using OpenAI's platform for either item generation or embedding. If NULL, users must provide either `groq.API` for item generation via Groq or `hf.token` for embeddings via Hugging Face.

`hf.token`
: A character string or NULL (optional, default: NULL). The Hugging Face API token for authentication with Hugging Face services. Required when using Hugging Face models for embeddings. If NULL, an `openai.API` key must be provided since the user will need to embed via OpenAI.

`main.prompts` A named list of character strings or NULL (optional, default: NULL). Custom prompts for item generation. If provided, this must be a named list where `names(main.prompts)` equals `names(item.attributes)`. Each prompt must explicitly mention all attributes found in the associated element of `item.attributes`. Users should not include instructions regarding layout/formatting of LLM response, as this is handled automatically for proper parsing. If NULL, AIGENIE builds appropriate prompts automatically based on other prompt-building parameters.

`groq.API` A character string or NULL (optional, default: NULL). The Groq API key for authentication with Groq's LLM services. Required when users want to generate items via Groq's API platform using open-source models. Commonly used in combination with `openai.API` since Groq does not provide embedding services.

`anthropic.API` A character string or NULL (optional, default: NULL). The Anthropic API key for authentication with Anthropic's Claude models. Required when using Claude models (e.g., "sonnet", "opus", "haiku") for item generation. Get a key at https://console.anthropic.com/.

`jina.API` A character string or NULL (optional, default: NULL). The Jina AI API key for authentication with Jina's embedding services. Required when using Jina embedding models (e.g., "jina-embeddings-v3", "jina-embeddings-v4"). Free tier available at https://jina.ai/.

`model` A character string (optional, default: "gpt4o"). Specifies which large language model to use for item generation. Supports models from multiple providers:

- **OpenAI**: `"gpt-4o"`, `"gpt-4"`, `"gpt-3.5-turbo"`, `"o1"`, `"o1-mini"`
- **Anthropic**: `"sonnet"`, `"opus"`, `"haiku"`, or full names like `"claude-sonnet-4-5-20250929"`
- **Groq**: `"llama-3.3-70b-versatile"`, `"mixtral-8x7b-32768"`, `"gemma2-9b-it"`, `"deepseek-r1-distill-llama-70b"`, `"qwen-2.5-72b"`

Aliases like `"llama"`, `"mixtral"`, `"gemma"`, `"deepseek"`, `"claude"` are also accepted. The function automatically determines which API service to use based on the model name and available API keys.

`temperature` A numeric value (optional, default: 1). Controls the randomness and creativity of the LLM's item generation. Must be between 0-2, where lower values produce more deterministic outputs and higher values increase creativity and variability.

`top.p` A numeric value (optional, default: 1). Controls nucleus sampling for the LLM's text generation. Must be between 0-1, where lower values make the model more focused and higher values allow more diverse outputs. Can be used in conjunction with `temperature`.

`embedding.model`
A character string (optional, default: "text-embedding-3-small"). Specifies which model to use for generating embeddings of items. Supports multiple providers:

- **OpenAI**: `"text-embedding-3-small"`, `"text-embedding-3-large"`, `"text-embedding-ada-002"`

- **Jina AI**: `"jina-embeddings-v3"`, `"jina-embeddings-v4"`, `"jina-embeddings-v2-base-en` (requires `jina.API`)
- **HuggingFace**: `"BAAI/bge-small-en-v1.5"`, `"BAAI/bge-base-en-v1.5"`, `"thenlper/gte-base"`, `"sentence-transformers/all-MiniLM-L6-v2"`

The provider is automatically detected based on the model name. Jina models support task adapters and Matryoshka dimension truncation for optimized embeddings.

`target.N` An integer, named list of integers, or NULL (optional, default: NULL). Specifies the number of items to generate for each item type. Can be a single integer (applies to all item types) or a named list of integers where `names(target.N)` equals `names(item.attributes)` for different numbers per item type. If NULL, 60 items per item type will be generated. A rule of thumb is about 60 items or more per item type for meaningful reduction analysis.

`domain` A character string or NULL (optional, default: NULL). Specifies the psychological or research domain for context in item generation. Should be specific (e.g., "personality", "child development") rather than general. If supplied, it will be used to construct appropriate prompts and system roles unless `system.role` is provided.

`scale.title` A character string or NULL (optional, default: NULL). Specifies the name or title of the scale being developed. Can be formal or descriptive, but more specific titles generally produce better results. If supplied, it will be used to construct appropriate prompts and system roles unless `system.role` is provided.

`item.examples` A data frame or NULL (optional, default: NULL). Provides example items to guide the LLM's generation style and format. Must be a data frame with columns: `statement` (the actual item), `attribute` (the item's attribute), and `type` (the item's type). All values must be non-empty strings, and the `attribute` and `type` must align with the `item.attributes` object. Items should be extremely high quality and validated if possible, as they serve as style templates.

`audience` A character string or NULL (optional, default: NULL). Specifies the target population for the scale being developed. Should be as specific as possible (e.g., "educated adults in rural America", "children with ASD in second grade") rather than general demographic categories. If supplied, it will be used to construct appropriate prompts and system roles unless `system.role` is provided.

`item.type.definitions`
A named list of character strings or NULL (optional, default: NULL). Provides definitions or descriptions of each item type for the LLM. Must be a named list where `names(item.type.definitions)` equals `names(item.attributes)`. Useful when constructs or item types are obscure or potentially ambiguous, helping the LLM understand the item type or construct in your specific context. If supplied, it will be used to construct appropriate prompts and system roles unless `system.role` is provided.

`response.options`
: A character vector or NULL (optional, default: NULL). Specifies the response scale labels for the generated items (e.g., c("agree", "neither agree nor disagree", "disagree")). These labels provide context for item writing but do not appear in the actual items themselves. If supplied, it will be used to construct appropriate system roles unless `system.role` is provided.

`prompt.notes`
: A named list of character strings, character string, or NULL (optional, default: NULL). Allows users to add custom instructions or context to the prompts. Can be a named list where `names(prompt.notes)` equals `names(item.attributes)` for different notes per item type, or a single string applied to all item types. These notes are appended at the end of constructed prompts, allowing users to add brief additional requirements (e.g., "All items MUST begin with the stem 'I am someone who...'") without creating entirely custom prompts. Should be brief; otherwise, users should consider using `main.prompts`.

`system.role`
: A character string or NULL (optional, default: NULL). Defines the system role/persona for the LLM during item generation. If not provided, one is built automatically based on prompt-building parameters. Should be as specific as possible (e.g., "You are an expert scale developer and psychometrician with extensive expertise in drafting Likert-type items for children with ASD. Today, you will focus on developing robust, single-statement items that assess linguistic ability."). Applies to all LLM interactions.

`EGA.model`
: A character string or NULL (optional, default: NULL). Specifies which model to use for Exploratory Graph Analysis network construction. Valid options are "tmfg" or "glasso". If NULL, AIGENIE will test both "tmfg" and "glasso" models and automatically return the model that maximizes NMI (normalized mutual information). TMFG is a greedy but speedy network-building algorithm that works well for many applications, especially text. EBICglasso is slower but non-greedy and may capture more nuanced relationships.

`EGA.algorithm`
: A character string (optional, default is "walktrap" when there is a single trait and "louvain" when there is more than one trait). Specifies which community detection algorithm to use within the EGA framework. Valid options are "louvain", "walktrap", or "leiden". The algorithm operates separately from the network building specified by `EGA.model`.

`EGA.uni.method`
: A character string (optional, default: "louvain"). Specifies the method for handling unidimensional structures in EGA. Valid options are: "expand" (expands correlation matrix with four variables correlated 0.50; if dimensions 2, data are unidimensional), "LE" (applies Leading Eigenvector algorithm; if dimensions = 1, uses LE solution), or "louvain" (applies Louvain algorithm; if dimensions = 1, uses Louvain solution). This parameter is rarely modified by users.

`uva.cut.off`
: A numeric value in `[0, 1)` (optional, default: `0.20`). The weighted topological overlap threshold passed to `EGAnet::UVA` during the redundancy-reduction step. Items with pairwise wTO at or above this value are flagged

as redundant. Lower values are more aggressive (remove more items); higher values are more conservative.

`keep.org` A logical value (optional, default: FALSE). Controls whether the pre-reduced items generated by the model are returned to the user. If TRUE, returns the full item pool before psychometric reduction. Does not affect the reduction process.

`items.only` A logical value (optional, default: FALSE). Controls whether the function only generates items without running the full psychometric pipeline. If TRUE, skips embedding, EGA, and reduction steps, returning only a data frame with columns `ID`, `statement`, `type`, and `attribute`. Useful when users want to generate items with AIGENIE, embed them elsewhere, and use the `GENIE` function for reduction.

`embeddings.only` A logical value (optional, default: FALSE). Controls whether the function generates items and embeddings but skips psychometric reduction. If TRUE, returns a named list with `embeddings` (the embedding matrix) and `items` (the items data frame). If both `items.only` and `embeddings.only` are TRUE, defaults to `embeddings.only` behavior.

`adaptive` A logical value (optional, default: TRUE). Controls whether previously generated items are incorporated into subsequent prompts to reduce redundancy. Items are generated in batches to avoid context window limitations, potentially requiring multiple API calls. When TRUE, appends previously generated items so the model knows what has been generated to avoid repetition. Should always be enabled unless context limitations are a concern.

`run.overall` A logical value (optional, default: FALSE). Controls whether a *fit* analysis on the complete item pool is run *post-reduction.* By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, an additional analysis is run on the overall sample, but no further reductions at the overall level are made. If only one item type is present, this argument will be ignored.

`all.together` A logical value (optional, default: FALSE). Controls whether the *reduction* analysis on the complete item pool is run. By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, reductions are made at the overall level (i.e., all items go through the reduction pipeline together, agnostic of item type). If only one item type is present, this argument will be ignored.

`plot` A logical value (optional, default: TRUE). Controls whether visualizations are generated and displayed. When TRUE, generates EGA network comparison plots (before vs after reduction) for each item type and the sample overall. Plots are always saved and returned in the output object but can be suppressed from display for cleaner output.

`silently` A logical value (optional, default: FALSE). Controls console output and messaging during function execution. When TRUE, suppresses progress statements about item generation, embedding, and pipeline reduction.

Does not affect warnings or errors, only informational messages. Operates independently of the `plot` parameter.

Value

The structure of the return value depends on the function flags.

Defaults: `items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `keep.org = FALSE`, `all.together = FALSE`.

When `items.only = TRUE`**:** Returns a `data.frame` of generated items with columns: `ID`, `statement`, `type`, and `attribute`.

When `embeddings.only = TRUE`**:** Returns a named `list` with two elements:

- `embeddings` — an embedding matrix/list (columns or rownames correspond to item IDs).
- `items` — the items `data.frame` described above.

Default behaviour (`items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `keep.org = FALSE`, `all.together = FALSE`): Returns a named `list` with two top-level elements:

`item_type_level` A named list where each name is an item type and each element is a per-type named list containing:

`final_NMI` Numeric: final normalized mutual information after reduction.

`initial_NMI` Numeric: initial NMI of the pre-reduced item pool.

`embeddings` List or matrix of embeddings for this item type (see 'Notes on `embeddings`' below).

`UVA` List from Unique Variable Analysis (contains at least `n_removed`, `n_sweeps`, `redundant_pairs` data.frame).

`bootEGA` List with bootEGA results (e.g. `initial_boot`, `final_boot`, `n_removed`, `items_removed`, `initial_boot_with_redundancies`).

`EGA.model_selected` Character: chosen EGA model (e.g. `"TMFG"` or `"Glasso"`).

`final_items data.frame`: final items after reduction (columns include `ID`, `statement`, `attribute`, `type`, `EGA_com`).

`final_EGA` EGA object (from EGAnet) after reduction.

`initial_EGA` Initial EGA object computed on the pre-reduced item set.

`start_N` Integer: initial number of items in this type.

`final_N` Integer: final number of items in this type.

`network_plot ggplot / patchwork` object comparing networks before vs after reduction.

`stability_plot ggplot / patchwork` object showing item stability before vs after reduction.

`overall` Named list with aggregated results across all item types. Under the default this contains:

`final_items data.frame` of final items across all types (columns as above).

`embeddings` Embeddings for the full reduced item set (see 'Notes on `embeddings`' below). Note: `overall$embeddings` does **not** include `selected`.

When `keep.org = TRUE` (in addition to defaults above): The top-level shape remains (`item_type_level` and `overall`) but includes original (pre-reduction) information:

`item_type_level` Each per-type sublist contains: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `initial_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, `stability_plot`.

`overall` Contains `final_items`, `initial_items`, and `embeddings` for the full item pool.

For `keep.org = TRUE`, per-type `embeddings` contains at least: `full_org`, `sparse_org`, `selected`, `full`, and `sparse`. (`overall$embeddings` contains the same subcomponents **except** `selected` is omitted.)

When `run.overall = TRUE` (`items.only = FALSE`, `embeddings.only = FALSE`):

`item_type_level` Same per-type structure as the default (see above).

`overall` A named list with aggregated results (not limited to `final_items` and `embeddings`) containing: `final_NMI`, `initial_NMI`, `embeddings`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, and `network_plot`.

When `all.together = TRUE` (regardless of `run.overall`): Results are **not** split into `item_type_level` and `overall`. Instead the function returns a single named list (applies to the full — possibly `keep.org` modified — result set) containing: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, and `stability_plot`.

References

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLOS ONE, 12*(6), e0174035. https://doi.org/10.1371/journal.pone.0174035

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research, 58*(6), 1165–1182. https://doi.org/10.1080/00273171.2023.2194606

Christensen, A. P., & Golino, H. (2021). Estimating the stability of psychological dimensions via bootstrap exploratory graph analysis: A Monte Carlo simulation and tutorial. *Psych, 3*(3), 479–500. https://doi.org/10.3390/psych3030032

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment, 2005*(9), P09008. https://doi.org/10.1088/1742-5468/2005/09/P09008

Russell-Lasalandra, L. L., Christensen, A. P., & Golino, H. (2024). Generative psychometrics via AI-GENIE: Automatic item generation and validation via network-integrated evaluation. https://osf.io/preprints/psyarxiv/fgbj4_v2.

Examples

```
## Not run:
#########################################################
#### Example 1: Using AI-GENIE with Default Prompts ####
#########################################################
```

```
# Add an OpenAI API key
key <- "INSERT YOUR KEY HERE"

# Item type definitions
trait.definitions <- list(
  neuroticism = "Neuroticism is a personality trait that describes one's tendency to experience negative
  openness = "Openness is a personality trait that describes how open-minded, creative, and imaginative a
  extraversion = "Extraversion is a personality trait that describes people who are more focused on the e
)

# Item attributes
aspects.of.personality.traits <- list(
  neuroticism = c("anxious", "depressed", "insecure", "emotional"),
  openness = c("creative", "perceptual", "curious", "philosophical"),
  extraversion = c("friendly", "positive", "assertive", "energetic")
)

# Name the field or specialty
domain <- "Personality Measurement"

# Name the Inventory being created
scale.title <- "Three of 'Big Five:' A Streamlined Personality Inventory"

# Run AI-GENIE to generate, validate, and redundancy-check an item pool for your new scale.
personality.inventory.results <- AIGENIE(
  item.attributes = aspects.of.personality.traits,
  openai.API = key,
  domain = domain,
  scale.title = scale.title,
  item.type.definitions = trait.definitions
)

# View the final item pool
View(personality.inventory.results)


##########################################################
#### Example 2: Using AI-GENIE with Custom Prompts ####
##########################################################


# Define a custom system role
system.role <- "You are an expert methodologist who specializes in scale development for personality meas

# Define custom prompts for each personality trait
custom.personality.prompts <- list(

  # Prompt for generating neuroticism traits
  neuroticism = paste0(
    "Generate unique, psychometrically robust single-statement items designed to assess ",
    "the Big Five personality trait neuroticism.",
    "Neuroticism has the following characteristics: anxious, depressed, insecure, and emotional. "
  ),
```

```
  # Prompt for generating openness traits
  openness = paste0(
    "Generate unique, psychometrically robust single-statement items designed to assess ",
    "the Big Five personality trait openness.",
    "Openness has the following characteristics: creative, perceptual, curious, and philosophical"
  ),

  # Prompt for generating extraversion traits
  extraversion = paste0(
    "Generate unique, psychometrically robust single-statement items designed to assess ",
    "the Big Five personality trait extraversion.",
    "Extraversion has the following characteristics: friendly, positive, assertive, and energetic."
  )

)

# Run AI-GENIE to generate, validate, and redundancy-check an item pool for your new scale.
personality.inventory.results.custom <- AIGENIE(
  item.attributes = aspects.of.personality.traits, # created in example 1
  main.prompts = custom.personality.prompts,
  system.role = system.role,
  openai.API = key, # created in example 1
  scale.title = scale.title # created in example 1
)

# View the final item pool
View(personality.inventory.results.custom)

#####################################################################
###### Or, Run AIGENIE with an Open Source Model via Groq ######
#####################################################################

# Add your API Key from Groq
groq.key <- "INSERT YOUR KEY HERE"

# Chose an open-source model like 'DeepSeek' or 'GPT oss'
open.source.model <- "GPT oss 120b"

# Use AIGENIE with an open source model via Groq
personality.inventory.results.gptoss <- AIGENIE(
  item.attributes = aspects.of.personality.traits, # created in example 1
  openai.API = key, # Created in example 1
  domain = domain, # Created in example 1
  scale.title = scale.title, # Created in example 1
  model = open.source.model, # Select a model available on Groq's API
  groq.API = groq.key
)

# View the final item pool
View(personality.inventory.results.gptoss)

#####################################################################
```

```
###### Or, Run AIGENIE with a Hugging Face Embedding Model #####
######################################################################

# Chose a BAAI/bge series OR thenlper/gte series model
hf.embedding.model <- "BAAI/bge-large-en-v1.5"

# Create a HF Token to access the best models. Moderate useage will still be FREE
hf.token <- "INSERT YOUR KEY HERE"


# Use AIGENIE with an open source model via Groq
personality.inventory.results.hf <- AIGENIE(
  item.attributes = aspects.of.personality.traits, # created in example 1
  # OpenAI API key is not needed for this example #
  domain = domain, # Created in example 1
  scale.title = scale.title, # Created in example 1
  model = open.source.model, # Select a model available on Groq's API
  groq.API = groq.key,
  embedding.model = hf.embedding.model,
  hf.token = hf.token
)

# View the final item pool
View(personality.inventory.results.hf)

######################################################################
#### Example 4: Using Anthropic Claude for Item Generation ####
######################################################################

# Add your Anthropic API key
anthropic.key <- "INSERT YOUR KEY HERE"

# Use Claude Sonnet (or "opus", "haiku", or full model names)
personality.inventory.claude <- AIGENIE(
  item.attributes = aspects.of.personality.traits,
  anthropic.API = anthropic.key,
  openai.API = key,  # Still needed for embeddings
  model = "sonnet",  # Alias for claude-sonnet-4-5-20250929
  domain = domain,
  scale.title = scale.title,
  item.type.definitions = trait.definitions
)

# View the final item pool
View(personality.inventory.claude)

######################################################################
#### Example 5: Using Jina AI Embeddings ####
######################################################################

# Add your Jina API key (free tier available)
jina.key <- "INSERT YOUR KEY HERE"
```

```
# Use Jina embeddings with Groq for generation
personality.inventory.jina <- AIGENIE(
  item.attributes = aspects.of.personality.traits,
  groq.API = groq.key,
  jina.API = jina.key,
  model = "llama-3.3-70b-versatile",
  embedding.model = "jina-embeddings-v3",
  domain = domain,
  scale.title = scale.title,
  item.type.definitions = trait.definitions
)

# View the final item pool
View(personality.inventory.jina)

#####################################################################
#### Example 6: Anthropic + Jina (No OpenAI Required) ####
#####################################################################

# Full pipeline without OpenAI
personality.inventory.no.openai <- AIGENIE(
  item.attributes = aspects.of.personality.traits,
  anthropic.API = anthropic.key,
  jina.API = jina.key,
  model = "sonnet",
  embedding.model = "jina-embeddings-v3",
  domain = domain,
  scale.title = scale.title,
  item.type.definitions = trait.definitions
)

# View the final item pool
View(personality.inventory.no.openai)


## End(Not run)
```

`build_item_attributes_from_items` *Build item.attributes Object from Items Data Frame*

Description

Reverse engineers the `item.attributes` object structure required by AIGENIE from a validated items data frame. This allows GENIE to work with user-provided items by reconstructing the expected attribute structure.

Usage

```
build_item_attributes_from_items(items)
```

Arguments

`items`	A validated data frame with columns: statement, attribute, type, ID (already processed by items_validate_GENIE)

Value

A named list where:

- Names are the unique item types from items$type
- Each element is a character vector of unique attributes for that type
- All values are normalized (lowercase, trimmed) to match AIGENIE expectations

---

`build_return` *Build the proper return object based on if the initial items should be kept and if an overall analysis was run*

Description

Build the proper return object based on if the initial items should be kept and if an overall analysis was run

Usage

```
build_return(item_type_level, overall_result, run.overall, keep.org)
```

Arguments

`item_type_level`	A named list containing the results at the item type level
`overall_result`	A named list containing the results at the overall level
`run.overall`	A flag saying if a fit analysis should be run on the items overall
`keep.org`	A flag saying if the original items generated should be kept

Value

A named list of lists containing the appropriately formated return object

`calc_final_stability` *Run bootstrapped EGA on the initial set of items*

Description

Run bootstrapped EGA on the initial set of items

Usage

```
calc_final_stability(
  result,
  data,
  EGA.algorithm,
  EGA.uni.method,
  corr = "auto",
  ncores = NULL,
  boot.iter = 100,
  silently,
  EGA.type = "EGA.fit"
)
```

Arguments

`result`	The running results of the item type level so far.
`data`	The embedding matrix to be used (either the sparse or full)
`EGA.algorithm`	The EGA algorithm to be used
`EGA.uni.method`	The EGA unidensinoality that should be used
`corr`	Character. Correlation method. Default ”auto”.
`ncores`	Numeric. Number of cores for parallel processing.
`boot.iter`	Numeric. Number of bootstrap iterations.
`silently`	A logical flag that decides whether to print output
`EGA.type`	Type of EGA (default ”EGA.fit”).

Value

An updated `results` object with the initial stability object added

`chat` *Chat with an LLM via API Calls*

Description

Send one or more prompts to a remote large-language model (LLM) using the appropriate provider API (OpenAI, Hugging Face, Groq, or Anthropic). A valid API key for at least one provider is required. To use a local model (no API call), see `local_chat()`.

Usage

```
chat(
  prompts,
  model,
  system.role = NULL,
  openai.API = NULL,
  hf.token = NULL,
  groq.API = NULL,
  anthropic.API = NULL,
  reps = 1,
  top.p = 1,
  temperature = 1,
  max.tokens = 2048L,
  silently = FALSE
)
```

Arguments

`prompts`	A character string or character vector. The main prompt(s) given to the model. If multiple prompts are supplied, each will be sent separately to the model.
`model`	A character string specifying the LLM model name (e.g., `"gpt4o"`). The model must correspond to the API key provided.
`system.role`	A character string or character vector, default `NULL`. The system role(s) (model persona). If only one system role is provided and multiple prompts are supplied, the same role will be used for each prompt. If multiple system roles are provided, they should align with the prompts.
`openai.API`	A character string, default `NULL`. Your OpenAI API key (required when using an OpenAI model).
`hf.token`	A character string, default `NULL`. Your Hugging Face token (required when using a Hugging Face-hosted model).
`groq.API`	A character string, default `NULL`. Your Groq API key (required when using a Groq-hosted model).
`anthropic.API`	A character string, default `NULL`. Your Anthropic API key (required when using an Anthropic model).

`reps`	Integer, default `1`. The number of times each prompt will be given to the model.
`top.p`	Numeric, default `1`. Top-p (nucleus) sampling parameter.
`temperature`	Numeric, default `1`. Sampling temperature controlling response randomness.
`max.tokens`	Integer, default `2048L`. Maximum number of tokens requested from the model.
`silently`	Logical, default `FALSE`. If `FALSE`, progress messages are printed. If `TRUE`, the function runs quietly.

Details

The function includes a retry mechanism (up to 5 attempts) for transient API failures. If all attempts fail, the function stops with an informative error.

Value

A `data.frame` with one row per API call (i.e., per prompt × repetition) containing:

- `rep` — repetition index
- `prompt` — the prompt text sent to the model
- `response` — the raw text response returned by the model

Important

This function requires a valid API key corresponding to the selected model. Network access is required. For local (non-API) models, use `local_chat()`.

See Also

`local_chat`

Examples

```
## Not run:
################################################
### Example 1: Writing a Very Basic Prompt  #####
################################################

# First, define your API key
key <- "INSERT YOUR KEY HERE"
# Then, define your LLM model. This model should correspond to your API key.
model <- "gpt4o" # in this example, you would need an OpenAI API key.

# Then, write your prompt. This will be given to the model directly.
prompt <- "Why does the planet Saturn have rings? Give a 100 word explanation."

# Optionally, add a system role (a model persona)
system.role <- "You specialize in tutoring astronomy for high school students."
```

```
# Add the number of prompt repetitions. By default, this is set to 1. But it
# may bu useful to increase the number of repetitions to get a sense of how
# consistent your output might be.
reps <- 3

# Now you are ready to chat with an LLM
first_chat <- chat(
  # Set your own API key. If you are not using OpenAI, change the 1st
  # argument to match your API key. Choices are `hf.token`, `groq.API`,
  # `anthropic.API`, and `openai.API`. In this example, I'm using
  # `openai.API` since I want to chat with a GPT model.
  openai.API = key,
  model = model, # Ensure your model corresponds to your API key.
  prompts = prompt,
  system.role = system.role,
  reps = reps
)

# Check how the output changes from iteration to iteration
first_chat$response[[1]] # first iteration output
first_chat$response[[2]] # second iteration output
first_chat$response[[2]] # third iteration output


######################################################################
### Example 2: Send multiple prompts in a single function call #####
######################################################################

# You are also able to send more than one prompt in a single call
# Let's pull the first prompt from Example 1:
prompt1 <- "Why does the planet Saturn have rings? Give a 100 word explanation."
prompt2 <- "Which planet is the hottest in our solar system? How do we know?"

# Aggregate the prompts in a single object
prompts <- c(prompt1, prompt2)

# Ask the model the questions
second_chat <- chat(
  openai.API = key, # defined in Example 1
  model = model, # defined in Example 1
  prompts = prompts, # NEW
  system.role = system.role, # defined in Example 1
  reps = reps # defined in Example 1
)

# The outputted data frame for this example will have 6 rows
# since the number of prompts (2) times the number of reps (3)
# gives a total of 6 API calls.
second_chat$response[second_chat$prompt==prompt1] # the responses from prompt 1
second_chat$response[second_chat$prompt==prompt2] # the responses from prompt 2


######################################################################
```

```
### Example 3: Send multiple prompts with different System Roles ###
#####################################################################

# Perhaps your prompts are not related. In that case, you would probably want
# to set a different system role for each prompt.
# Let's change `prompt 2` to be entirely unrelated to astronomy.
prompt2 <- "What is the difference between eukaryotes and prokaryotes? Why?"

# This new second prompt does not fit with the astronomy tutor persona. Let's
# write a persona to match this new prompt topic.
system.role2 <- "You specialize in tutoring biology for middle school students."

# Now, let's combine the system roles into a single object
system.role <- c(system.role, # defined in Example 1: the astronomy tutor
                 system.role2 # defined above: the biology tutor
                 )

# Aggregate our prompts in a single object again
prompts <- c(prompt1, # Asks about Saturn's rings (needs astronomy tutor)
             prompt2 # Asks about types of cells (needs biology tutor)
             )

# Ask the model the questions
third_chat <- chat(
  openai.API = key, # defined in Example 1
  model = model, # defined in Example 1
  prompts = prompts, # NEW
  system.role = system.role, # NEW
  reps = reps # defined in Example 1
)

# View the outputted data frame to examine the responses
View(third_chat)

## End(Not run)
```

`check_for_default_APIs` *Check for users who pasted the example code but didn't add an API key*

Description

Check for users who pasted the example code but didn't add an API key

Usage

```
check_for_default_APIs(
  hf.token,
```

```
  groq.API = NULL,
  openai.API,
  anthropic.API = NULL,
  jina.API = NULL
)
```

Arguments

hf.token	The hugging face token provided
groq.API	The Groq API key provided
openai.API	The OpenAI API key provided

---

check_local_llm_setup

Check Local LLM Setup

Description

Verifies that all requirements for local LLM inference are met, including Python environment, llama-cpp-python installation, and model file accessibility.

Usage

```
check_local_llm_setup(model.path, silently = FALSE)
```

Arguments

model.path	Path to the GGUF model file
silently	Logical. Suppress progress messages?

Value

Logical. TRUE if setup is complete, FALSE otherwise.

---

embedding_matrix_validate_GENIE

Validate Embedding Matrix for GENIE

Description

Validates that the optional embedding matrix meets all requirements for GENIE processing. Ensures proper structure, dimensions, column names match item IDs, and numeric content.

Usage

```
embedding_matrix_validate_GENIE(embedding.matrix, items, silently = FALSE)
```

Arguments

`embedding.matrix`	A numeric matrix or data frame with rows as embedding dimensions and columns as items. Can be NULL if embeddings will be generated.
`items`	A validated items data frame (already processed by items_validate_GENIE)
`silently`	Logical. If FALSE, displays informational messages

Value

A validated embedding matrix (always as matrix type) or NULL if not provided

---

`ensure_aigenie_python` *Ensure AI-GENIE Python Environment is Ready*

Description

Sets up the Python environment with all required dependencies for AI-GENIE. Uses UV for fast, reliable package management. This function is called automatically when needed, but can also be called directly.

Usage

```
ensure_aigenie_python(
  force_reinstall = FALSE,
  include_huggingface = TRUE,
  include_local_llm = FALSE,
  gpu = FALSE
)
```

Arguments

`force_reinstall`	Logical. Force complete reinstallation?
`include_huggingface`	Logical. Include HuggingFace packages? Default TRUE.
`include_local_llm`	Logical. Include local LLM support? Default FALSE.
`gpu`	Logical. Install GPU-enabled PyTorch? Default FALSE.

Value

TRUE invisibly on success

`final_community_detection` *Run Final Community Detection with EGA*

Description

Run Final Community Detection with EGA

Usage

```
final_community_detection(
  embedding_matrix,
  true_communities,
  model = "glasso",
  algorithm = "walktrap",
  uni.method = "louvain",
  corr = "auto"
)
```

Arguments

`embedding_matrix`	A numeric matrix with items as columns.
`true_communities`	Named list mapping items to known communities.
`model`	Network estimation model (e.g., "glasso", "TMFG").
`algorithm`	Community detection algorithm (e.g., "walktrap").
`uni.method`	Unidimensionality method passed to EGA.
`corr`	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.

Value

A list with final communities, final NMI, dropped items, EGA object, and success flag.

`GENIE` *The use of the psychometric reduction component of AIGENIE on your pre-existing item pool*

Description

GENIE applies the psychometric reduction steps present in `AIGENIE` on user-supplied items. Users provide their own items and optionally their own embeddings, then `GENIE` performs redundancy reduction and structural validation to assess item quality and dimensionality.

Usage

```
GENIE(
  items,
  embedding.matrix = NULL,
  openai.API = NULL,
  hf.token = NULL,
  jina.API = NULL,
  embedding.model = "text-embedding-3-small",
  EGA.model = NULL,
  EGA.algorithm = NULL,
  EGA.uni.method = NULL,
  uva.cut.off = 0.2,
  embeddings.only = FALSE,
  run.overall = FALSE,
  all.together = FALSE,
  plot = TRUE,
  silently = FALSE
)
```

Arguments

`items` Data frame with columns: statement, attribute, type, ID. All columns must be character type except ID (numeric or character allowed).

- `statement`: The actual item text
- `attribute`: The construct/attribute the item measures
- `type`: The item type/category
- `ID`: Unique identifier for each item

`embedding.matrix` Optional numeric matrix or data frame where:

- Rows represent embedding dimensions
- Columns represent items (must match items$ID exactly)
- If `NULL`, embeddings will be generated using `embedding.model`

`openai.API` OpenAI API key (required if using OpenAI embedding models)

`hf.token` HuggingFace token (optional, improves rate limits for HF models)

`jina.API` Jina AI API key for using Jina embedding models (e.g., "jina-embeddings-v3"). Free tier available at https://jina.ai/.

`embedding.model` Embedding model to use if embedding.matrix not provided:

- OpenAI: "text-embedding-3-small", "text-embedding-3-large", "text-embedding-ada-002"
- Jina AI: "jina-embeddings-v3", "jina-embeddings-v4", "jina-embeddings-v2-base-en" (requires jina.API)
- HuggingFace: "BAAI/bge-base-en-v1.5", "BAAI/bge-small-en-v1.5", "sentence-transformers/all-MiniLM-L6-v2"

`EGA.model` EGA network estimation model ("glasso", "TMFG", or NULL for auto-selection)

`EGA.algorithm` EGA community detection algorithm ("walktrap", "leiden", "louvain")

`EGA.uni.method`
Unidimensionality assessment method ("louvain", "expand", "LE")

`uva.cut.off` Numeric in `[0, 1)`. wTO threshold passed to `EGAnet::UVA` for the redundancy-reduction step (default: 0.20). Lower values remove more items.

`embeddings.only`
If `TRUE`, return embeddings and stop (skip network analysis)

`run.overall` A logical value (optional, default: FALSE). Controls whether a *fit* analysis on the complete item pool is run *post-reduction.* By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, an additional analysis is run on the overall sample, but no further reductions at the overall level are made. If only one item type is present, this argument will be ignored.

`all.together` A logical value (optional, default: FALSE). Controls whether the *reduction* analysis on the complete item pool is run. By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, reductions are made at the overall level (i.e., all items go through the reduction pipeline together, agnostic of item type). If only one item type is present, this argument will be ignored.

`plot` If `TRUE`, display network comparison plots

`silently` If `TRUE`, suppress progress messages

`model` Language model identifier (currently unused in GENIE)

Value

The structure of the return value depends on the function flags.

Defaults: `items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `all.together = FALSE`.

When `items.only = TRUE`**:** Returns a `data.frame` of generated items with columns: `ID`, `statement`, `type`, and `attribute`.

When `embeddings.only = TRUE`**:** Returns a named `list` with two elements:

- `embeddings` — an embedding matrix/list (columns or rownames correspond to item IDs).
- `items` — the items `data.frame` described above.

Default behaviour (`items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `all.together = FALSE`): Returns a named `list` with two top-level elements:

`item_type_level` A named list where each name is an item type and each element is a per-type named list containing:

`final_NMI` Numeric: final normalized mutual information after reduction.

`initial_NMI` Numeric: initial NMI of the pre-reduced item pool.

`embeddings` List or matrix of embeddings for this item type (see 'Notes on `embeddings`' below).

`UVA` List from Unique Variable Analysis (contains at least `n_removed`, `n_sweeps`, `redundant_pairs` data.frame).

`bootEGA` List with bootEGA results (e.g. `initial_boot`, `final_boot`, `n_removed`, `items_removed`, `initial_boot_with_redundancies`).

`EGA.model_selected` Character: chosen EGA model (e.g. `"TMFG"` or `"Glasso"`).

`final_items data.frame`: final items after reduction (columns include `ID`, `statement`, `attribute`, `type`, `EGA_com`).

`final_EGA` EGA object (from EGAnet) after reduction.

`initial_EGA` Initial EGA object computed on the pre-reduced item set.

`start_N` Integer: initial number of items in this type.

`final_N` Integer: final number of items in this type.

`network_plot ggplot` / `patchwork` object comparing networks before vs after reduction.

`stability_plot ggplot` / `patchwork` object showing item stability before vs after reduction.

`overall` Named list with aggregated results across all item types. Under the default this contains:

`final_items data.frame` of final items across all types (columns as above).

`embeddings` Embeddings for the full reduced item set (see 'Notes on `embeddings`' below). Note: `overall$embeddings` does **not** include `selected`.

When `run.overall = TRUE` (`items.only = FALSE`, `embeddings.only = FALSE`):

`item_type_level` Same per-type structure as the default (see above).

`overall` A named list with aggregated results (not limited to `final_items` and `embeddings`) containing: `final_NMI`, `initial_NMI`, `embeddings`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, and `network_plot`.

When `all.together = TRUE` (regardless of `run.overall`): Results are **not** split into `item_type_level` and `overall`. Instead the function returns a single named list containing: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, and `stability_plot`.

References

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLOS ONE, 12*(6), e0174035. https://doi.org/10.1371/journal.pone.0174035

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research, 58*(6), 1165–1182. https://doi.org/10.1080/00273171.2023.2194606

Christensen, A. P., & Golino, H. (2021). Estimating the stability of psychological dimensions via bootstrap exploratory graph analysis: A Monte Carlo simulation and tutorial. *Psych, 3*(3), 479–500. https://doi.org/10.3390/psych3030032

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment, 2005*(9), P09008. https://doi.org/10.1088/1742-5468/2005/09/P09008

Russell-Lasalandra, L. L., Christensen, A. P., & Golino, H. (2024). Generative psychometrics via AI-GENIE: Automatic item generation and validation via network-integrated evaluation. https://osf.io/preprints/psyarxiv/fgbj4_v2.

Examples

```
 ## Not run:
#############################################################
#### Using GENIE with OpenAI's Embeddings (Recommended) ####
#############################################################

# Add an OpenAI API Key
key <- "INSERT YOUR KEY HERE"


# Specify item statements that you already have written
statements <- c(
  "I find myself naturally initiating conversations with strangers at social gatherings.",
  "I enjoy creating a welcoming atmosphere for people I meet for the first time.",
  "I generally maintain a hopeful outlook, even when faced with challenges.",
  "I frequently find myself in a good mood, spreading cheer to those around me.",
  "I often have the drive to engage in exciting activities, even after a long day.",
  "I tend to tackle projects with enthusiasm and high energy from start to finish.",
  "I actively seek to include others in group activities, making them feel part of the team.",
  "I frequently reach out to new acquaintances to foster connections and friendships.",
  "I habitually focus on the silver lining in difficult situations, maintaining an optimistic perspective
  "I often express gratitude for the positive aspects of my life, which enhances my overall mood.",
  "I find joy in taking on new challenges that require a burst of energy and enthusiasm.",
  "I thrive in dynamic environments that keep me on my toes and invigorate my spirit.",
  "I take pleasure in introducing people to one another, acting as a social connector.",
  "I enjoy making others comfortable by engaging them in light-hearted conversation.",
  "I often set a positive tone in group settings with my upbeat demeanor.",
 "I approach each day with a sense of excitement and a positive mindset.",
 "I am drawn to fast-paced environments where I can express my high energy levels.",
 "I feel invigorated when working on multiple projects that demand my full attention.",
 "I take delight in meeting new people and quickly making them feel at ease.",
 "I find it rewarding to help shy or reserved individuals become involved in group discussions.",
 "I have a natural tendency to uplift others with my positive remarks and outlook.",
 "I find happiness in highlighting the successes of others, contributing to a cheerful environment.",
 "I eagerly immerse myself in activities that demand stamina and sustained energy.",
 "I often channel my vitality into hobbies and sports that require physical exertion.",
 "I feel rejuvenated when I bring people together to collaborate and share ideas.",
 "I often extend a genuine greeting to others, creating an inviting atmosphere.",
 "I regularly see challenges as opportunities for growth and learning.",
 "I commonly radiate positivity, influencing the mood of those around me.",
 "I approach mornings with anticipation and vigor, ready to embrace the day.",
 "I consistently infuse enthusiasm into group activities, boosting collective energy levels.",
 "I make an effort to connect with people by remembering details about their lives.",
 "I genuinely enjoy learning about people's diverse experiences and viewpoints.",
```

```
"I have a habit of encouraging others to see the bright side of their situations.",
"I believe in celebrating small victories, finding joy in daily accomplishments.",
"I often find myself eager to start the day with ambitious plans and goals.",
"I am known for sustaining high levels of energy during extended work sessions or projects.",
"I make an effort to engage those around me in meaningful and enjoyable conversations.",
"I often seek opportunities to bring people together, fostering a sense of community.",
"I naturally inspire others with my optimistic outlook, even in uncertain times.",
"I frequently look for the positive aspects in challenging situations and share them with others.",
"I approach new experiences with an eagerness and fervor that motivates those around me.",
"I thrive on maintaining high energy levels throughout demanding and fast-paced days.",
"I take pleasure in initiating warm interactions in group settings to make everyone comfortable.",
"I enjoy hosting gatherings that connect friends and encourage social bonding.",
"I am skilled at turning setbacks into learning experiences to maintain a positive outlook.",
"I always try to highlight the benefits in situations, enhancing a cheerful atmosphere.",
"I find excitement in starting the day with a list of activities to energize my routine.",
"I relish the challenge of keeping up with dynamic schedules that require sustained energy.",
"I often find joy in making newcomers feel welcome and appreciated in group settings.",
"I genuinely enjoy striking up conversations to learn more about the people I encounter.",
"I have a knack for seeing potential in situations that others might overlook.",
"I consistently try to uplift the mood in my surroundings with hopeful and encouraging words.",
"I frequently harness my energy to inspire and motivate those around me in team environments.",
"I often feel invigorated by challenges that require sustained focus and dynamic thinking.",
"I often create environments where people feel encouraged to share their thoughts freely.",
"I find it fulfilling to engage deeply with people, building lasting connections.",
"I see potential in every day, believing it holds opportunities for something good.",
"I actively focus on the pleasures of life, which naturally enhances my mood.",
"I am invigorated by opportunities to engage in lively and spirited events.",
"I tend to maintain momentum throughout the day, sustaining my energy levels.",
"I frequently experience sudden shifts in my emotions even when there is no apparent reason.",
"People often find it difficult to predict my emotional reactions to different situations.",
"I often doubt my abilities and worry about whether I am meeting expectations.",
"I frequently question my self-worth and tend to seek reassurance from others.",
"I become annoyed easily over small inconveniences or delays.",
"I often find myself feeling agitated or frustrated in situations that don't bother most people.",
"My mood can change drastically over the course of a day, often without any clear reason.",
"I tend to experience emotional highs and lows more intensely than those around me.",
"I sometimes avoid taking on new challenges because I fear not being good enough.",
"I often feel uncertain about my social standing and worry about being accepted by others.",
"I find myself getting irritated quickly when things don't go my way.",
"Minor annoyances often cause my patience to wear thin unusually fast.",
"I frequently struggle to maintain a stable emotional state throughout the day.",
"Unexpected events can cause me to experience drastic emotional swings.",
"I often feel inadequate in comparison to others around me.",
"I tend to second-guess my choices due to a lack of confidence in myself.",
"I tend to become frustrated when things do not proceed as I have planned.",
"I am prone to irritation when faced with unexpected changes to my routine.",
"My emotional state is often unpredictable, shifting from contentment to sadness with little warning.",
"People have commented that my emotions seem to fluctuate more than those of others.",
"I frequently feel self-conscious about my achievements compared to those of my peers.",
"I often worry excessively about making mistakes, even in situations where it might be inconsequential."
"Small disruptions in my daily routine can trigger strong feelings of annoyance.",
"I find myself becoming irritated more quickly than others when under stress or pressure.",
"I often find my emotional responses to be unpredictable, feeling fine one moment and unsettled the next
```

```
 "I experience strong emotions that can shift unexpectedly, often catching me off guard.",
 "I regularly feel uncertain about my ability to manage new responsibilities effectively.",
 "I often question my decisions, fearing they might not lead to the best outcomes.",
 "I frequently find myself reacting with impatience to situations perceived as minor interruptions.",
 "Even minor provocations can sometimes lead to an exaggerated sense of annoyance for me.",
 "My emotional state is often inconsistent, and I can feel ecstatic or despondent within short timeframes
 "I notice that my feelings can be quite volatile and intense, affecting how I interact with others throu
 "I regularly doubt whether I am capable of achieving my personal or professional goals.",
 "I often seek validation from others to feel reassured about my self-worth.",
 "I am sensitive to disturbances and find my patience wearing thin quickly when things aren't orderly.",
 "I occasionally struggle to contain my annoyance over trivial issues that disrupt my sense of calm.",
 "I can go from feeling upbeat to being downcast without an obvious cause.",
 "My emotional responses can sometimes be unpredictable, shifting with little notice.",
 "I often feel the need for affirmation about my abilities from friends or colleagues.",
 "I tend to compare myself to others and feel uncertain about my achievements.",
 "I find myself easily bothered by noises or disturbances in my environment.",
 "I get easily flustered by situations that interrupt my planned activities.",
 "I find it challenging to maintain a consistent emotional state, regardless of external situations.",
 "My emotional reactions can be intense and differ significantly from moment to moment.",
 "I have a persistent fear of not measuring up to the expectations placed on me.",
 "I often feel anxious about others' perceptions of my capabilities and appearance.",
 "I am quick to express frustration at minor inconveniences in my daily routine.",
 "I find that small, unforeseen events often disrupt my sense of calm and lead to irritation.",
 "My emotional reactions can be strong and relentless, impacting my behavior throughout the day.",
 "I often find myself emotionally labile, with an inner turbulence that others rarely perceive.",
 "I frequently worry about my competence in areas where others seem confident.",
 "I have a tendency to second-guess myself and require affirmation to feel reassured about my choices.",
 "Small disruptions can ignite a lingering sense of agitation within me.",
 "I often catch myself feeling irritable even in relatively calm settings.",
 "I find myself swinging from happy to melancholic in a short span of time, often surprising even myself.
 "Others often comment on how quickly my mood can change in response to seemingly minor events.",
 "I tend to feel apprehensive about presenting my opinions, fearing they may be judged harshly.",
 "I often require reassurance from peers to feel confident in my decisions and ideas.",
 "Interruptions during focused tasks often lead to an outpour of irritation from me.",
 "I struggle to keep my frustration in check when things do not unfold as expected."
)


# Create the item type and attribute labels
item.attributes <- c(
 rep(c("friendly", "positive", "energetic"), each = 2, times = 10),
 rep(c("moody", "insecure", "irritable"), each = 2, times = 10)
)
item.types <- c(
 rep("extraversion", 60),
 rep("neuroticism", 60)
)




# Build your data frame with the required columns: ID, statement, attribute, and type
items_df <- data.frame(
```

```
 ID = rep(as.factor(1:length(statements))),
 statement = statements,
 attribute = item.attributes,
 type = item.types
)


# Run GENIE with items you provide (embedding items via OpenAI)
example_reduction <- GENIE(items = items_df,
                           openai.API = key)

# View the results
View(example_reduction)


#####################################################################
###### Or, Run GENIE with a Hugging Face Embedding Model #######
#####################################################################

# Chose a BAAI/bge series OR thenlper/gte series model
hf.embedding.model <- "BAAI/bge-large-en-v1.5"

# Create a HF Token to access the best models. Moderate useage will still be FREE
hf.token <- "INSERT YOUR KEY HERE"

# Run GENIE using the Hugging Face Embedding model
example_reduction_HF <- GENIE(items = items_df,
                              embedding.model = hf.embedding.model,
                              hf.token = hf.token)




## End(Not run)
```

`get_local_llm` *Download a Local LLM Model*

Description

Downloads a GGUF model from HuggingFace for use with local_AIGENIE. Models are saved to a user-specified directory or the default AIGENIE models directory.

Usage

```
get_local_llm(repo_id, filename, save_dir = NULL, hf.token = NULL)
```

Arguments

`repo_id`	HuggingFace repository ID (e.g., "TheBloke/Mistral-7B-Instruct-v0.2-GGUF")

filename	Specific GGUF filename to download (e.g., "mistral-7b-instruct-v0.2.Q4_K_M.gguf")
save_dir	Directory to save the model. If NULL, uses the default AIGENIE models directory.
hf.token	Optional HuggingFace token for gated models

Value

Character string with the full path to the downloaded model file.

Examples

```
## Not run:
# Download a Mistral 7B model
model_path <- get_local_llm(
  repo_id = "TheBloke/Mistral-7B-Instruct-v0.2-GGUF",
  filename = "mistral-7b-instruct-v0.2.Q4_K_M.gguf"
)

# Use it with local_AIGENIE
results <- local_AIGENIE(
  item.attributes = my_attributes,
  model.path = model_path
)

## End(Not run)
```

---

install_gpu_support *Install GPU Support for AI-GENIE*

Description

Reinstalls the Python environment with GPU-enabled PyTorch for faster inference with local embedding models. Requires a CUDA-compatible NVIDIA GPU and proper driver installation.

Usage

```
install_gpu_support()
```

Details

This function:

1. Removes the existing Python environment
2. Creates a new environment with GPU-enabled PyTorch
3. Installs all HuggingFace dependencies

On Apple Silicon Macs, MPS (Metal Performance Shaders) acceleration is used automatically without needing this function.

Value

Invisible `TRUE` on success.

See Also

`reinstall_python_env`, `python_env_info`.

Examples

```
## Not run:
# Enable GPU acceleration (requires NVIDIA GPU + CUDA)
install_gpu_support()

## End(Not run)
```

---

install_local_llm_support

Install Local LLM Support

Description

Installs llama-cpp-python for running local GGUF models with `local_AIGENIE`. On Apple Silicon Macs, this includes Metal acceleration support for fast inference.

Usage

```
install_local_llm_support()
```

Details

After installation, you can use any GGUF model file with `local_AIGENIE()`. Download GGUF models from HuggingFace (search for "GGUF" format).

Popular model recommendations:

- **Llama 3 8B**: Good balance of quality and speed
- **Mistral 7B**: Fast with good quality
- **Qwen 2.5**: Strong multilingual support

Value

Invisible `TRUE` on success.

See Also

`local_AIGENIE`, `reinstall_python_env`.

Examples

```
## Not run:
# Install local LLM support
install_local_llm_support()

# Then download a GGUF model and use with local_AIGENIE
results <- local_AIGENIE(
  item.attributes = my_traits,
  model.path = "~/models/llama-3-8b.Q4_K_M.gguf",
  embedding.model = "bert-base-uncased"
)

## End(Not run)
```

---

item.examples_validate *Validate and Clean* `item.examples` *Against Cleaned* `items.attributes`

Description

Ensures `item.examples` is a data frame with required string columns and that the values in `type` and `attribute` align with the cleaned structure of `items.attributes`. Returns a cleaned version of the data frame with normalized values:

- `type` and `attribute` are trimmed and lowercased
- `statement` is trimmed (case preserved)

Usage

```
item.examples_validate(item.examples, items.attributes)
```

Arguments

`item.examples`	A data frame with columns `type`, `attribute`, `statement`. All values must be non-empty strings.
`items.attributes`	A cleaned list from `validate_items.attributes()`. All names and values must be normalized (lowercased and trimmed).

Value

A cleaned version of `item.examples` with normalized values.

`item.type.definitions_validate`
Validate and Clean `item.type.definitions`

Description

Validates that `item.type.definitions` is a named list where:

- Names are unique (after trim + case-fold)
- Names exist in `items.attributes`
- Values are non-empty strings

Usage

```
item.type.definitions_validate(item.type.definitions, items.attributes)
```

Arguments

`item.type.definitions`
: A named list of strings, where each name must correspond to a name in `items.attributes` and each value must be a non-empty string.

`items.attributes`
: A cleaned list from `validate_items.attributes()`.

Details

Returns a cleaned version with:

- Normalized names (trimmed and lowercased)
- Trimmed values (case preserved)

Value

A cleaned version of `item.type.definitions`.

`items.attributes_validate`
Validate `items.attributes`

Description

Validates that `items.attributes` is a **named list** whose names are truly unique after trimming whitespace and ignoring case, and that each element is itself a list **containing only strings**, with **at least two** truly unique strings (same trimming + case-insensitive rule).

Usage

```
items.attributes_validate(items.attributes)
```

Arguments

items.attributes	A named list. Each element must be a list containing only character scalars (strings). Each of those inner lists must contain at least two truly unique strings after trimming and case-folding.

Value

A cleaned version of `items.attributes` with normalized names and values. Errors are thrown if validation fails.

---

items_validate_GENIE *Validate Items Data Frame for GENIE*

Description

Validates that the items data frame meets all requirements for GENIE processing. Ensures proper structure, column presence, data types, and content validity.

Usage

```
items_validate_GENIE(items)
```

Arguments

items	A data frame that should contain columns: statement, attribute, type, ID

Value

A cleaned and validated items data frame with standardized formatting

`iterative_stability_check` *Iteratively run BootEGA to ensure structural stability of items*

Description

Iteratively run BootEGA to ensure structural stability of items

Usage

```
iterative_stability_check(
  embedding_matrix,
  items,
  cut.off = 0.75,
  model = "NULL",
  algorithm = "",
  uni.method,
  corr = "auto",
  ncores = NULL,
  boot.iter = 100,
  EGA.type = "EGA.fit",
  silently
)
```

Arguments

`embedding_matrix`	Numeric matrix of item embeddings (columns = items).
`items`	Data frame containing at least `ID` and `statement`.
`cut.off`	Numeric. Minimum stability required to retain an item.
`model`	Network estimation model (e.g., "glasso", "TMFG").
`algorithm`	Community detection algorithm.
`uni.method`	Unidimensionality method.
`corr`	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.
`ncores`	Numeric. Number of cores for parallel processing. Default NULL uses EGAnet default.
`boot.iter`	Numeric. Number of bootstrap iterations. Default 100.
`EGA.type`	Type of EGA (default "EGA.fit").
`silently`	Logical. Suppress output.

Value

A list containing the final embedding, boot objects, items removed per iteration, etc.

`list_available_models` *List Available Models*

Description

Queries the OpenAI, Groq, Anthropic, and/or Jina AI APIs to retrieve currently available models. Requires API keys for live provider queries. Jina AI models are returned from a curated static list (no list endpoint).

Usage

```
list_available_models(
  provider = NULL,
  openai.API = NULL,
  groq.API = NULL,
  anthropic.API = NULL,
  type = NULL
)
```

Arguments

`provider`	Optional. Filter by provider: "openai", "groq", "anthropic", "jina", or NULL for all.
`openai.API`	Optional OpenAI API key. If NULL, checks OPENAI_API_KEY env var.
`groq.API`	Optional Groq API key. If NULL, checks GROQ_API_KEY env var.
`anthropic.API`	Optional Anthropic API key. If NULL, checks ANTHROPIC_API_KEY env var.
`type`	Filter by model type: "chat", "embedding", or NULL for all. Default is NULL (show everything).

Value

A data frame with columns: provider, model, type, display_name, created

`local_AIGENIE` *Generate and Validate Psychometric Scale Items Using Local Models*

Description

Local version of AI-GENIE that uses locally installed language models and embeddings for complete privacy and offline operation. Generates items, creates embeddings, and performs network psychometric reduction entirely on the user's machine.

Usage

```
local_AIGENIE(
  item.attributes,
  model.path,
  embedding.model = "bert-base-uncased",
  main.prompts = NULL,
  temperature = 1,
  top.p = 1,
  target.N = NULL,
  domain = NULL,
  scale.title = NULL,
  item.examples = NULL,
  audience = NULL,
  item.type.definitions = NULL,
  response.options = NULL,
  prompt.notes = NULL,
  system.role = NULL,
  EGA.model = NULL,
  EGA.algorithm = NULL,
  EGA.uni.method = NULL,
  uva.cut.off = 0.2,
  n.ctx = 4096,
  n.gpu.layers = -1,
  max.tokens = 1024,
  device = "auto",
  batch.size = 32,
  pooling.strategy = "mean",
  max.length = 512L,
  keep.org = FALSE,
  items.only = FALSE,
  embeddings.only = FALSE,
  adaptive = TRUE,
  run.overall = FALSE,
  all.together = FALSE,
  plot = TRUE,
  silently = FALSE
)
```

Arguments

`item.attributes`	Named list of item types and their attributes (required)
`model.path`	Path to local GGUF model file (required)
`embedding.model`	Name or path to local embedding model (default: "bert-base-uncased")
`main.prompts`	Custom prompts for item generation (optional)
`temperature`	LLM temperature for randomness (0-2, default: 1)

`top.p` Top-p nucleus sampling parameter (0-1, default: 1)

`target.N` Number of items to generate per type (default: 60)

`domain` Content domain (e.g., "psychological")

`scale.title` Name of the scale

`item.examples` Data frame of example items

`audience` Target population

`item.type.definitions`
Definitions for item types

`response.options`
Response scale labels

`prompt.notes` Additional instructions for generation

`system.role` Custom system prompt

`EGA.model` Network model ("glasso", "TMFG", or NULL for auto)

`EGA.algorithm` Community detection algorithm (default: "walktrap" when there is one trait and "louvain" when there are multiple)

`EGA.uni.method`
Unidimensionality method (default: "louvain")

`uva.cut.off` Numeric in `[0, 1)`. wTO threshold passed to `EGAnet::UVA` for the redundancy-reduction step (default: 0.20). Lower values remove more items.

`n.ctx` Context window size (default: 4096)

`n.gpu.layers` GPU layers to use (-1 for all, default: -1)

`max.tokens` Maximum tokens per generation (default: 1024)

`device` Device for embeddings ("auto", "cpu", "cuda", "mps")

`batch.size` Batch size for embeddings (default: 32)

`pooling.strategy`
Pooling for embeddings ("mean", "cls", "max")

`max.length` Max sequence length for embeddings (default: 512)

`keep.org` Keep original items and embeddings (default: FALSE)

`items.only` Generate items only, skip reduction (default: FALSE)

`embeddings.only`
Generate embeddings only (default: FALSE)

`adaptive` Use adaptive generation (default: TRUE)

`run.overall` A logical value (optional, default: FALSE). Controls whether a *fit* analysis on the complete item pool is run *post-reduction*. By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, an additional analysis is run on the overall sample, but no further reductions at the overall level are made. If only one item type is present, this argument will be ignored.

`all.together` A logical value (optional, default: FALSE). Controls whether the *reduction* analysis on the complete item pool is run. By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, reductions are made at the overall level (i.e., all items go through the reduction pipeline together, agnostic of item type). If only one item type is present, this argument will be ignored.

`plot` Display network plots (default: TRUE)

`silently` Suppress progress messages (default: FALSE)

Value

The structure of the return value depends on the function flags.

Defaults: `items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `keep.org = FALSE`, `all.together = FALSE`.

When `items.only = TRUE`**:** Returns a `data.frame` of generated items with columns: `ID`, `statement`, `type`, and `attribute`.

When `embeddings.only = TRUE`**:** Returns a named `list` with two elements:

- `embeddings` — an embedding matrix/list (columns or rownames correspond to item IDs).
- `items` — the items `data.frame` described above.

Default behaviour (`items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `keep.org = FALSE`, `all.together = FALSE`): Returns a named `list` with two top-level elements:

`item_type_level` A named list where each name is an item type and each element is a per-type named list containing:

`final_NMI` Numeric: final normalized mutual information after reduction.

`initial_NMI` Numeric: initial NMI of the pre-reduced item pool.

`embeddings` List or matrix of embeddings for this item type (see 'Notes on `embeddings`' below).

`UVA` List from Unique Variable Analysis (contains at least `n_removed`, `n_sweeps`, `redundant_pairs` data.frame).

`bootEGA` List with bootEGA results (e.g. `initial_boot`, `final_boot`, `n_removed`, `items_removed`, `initial_boot_with_redundancies`).

`EGA.model_selected` Character: chosen EGA model (e.g. `"TMFG"` or `"Glasso"`).

`final_items` `data.frame`: final items after reduction (columns include `ID`, `statement`, `attribute`, `type`, `EGA_com`).

`final_EGA` EGA object (from EGAnet) after reduction.

`initial_EGA` Initial EGA object computed on the pre-reduced item set.

`start_N` Integer: initial number of items in this type.

`final_N` Integer: final number of items in this type.

`network_plot` `ggplot` / `patchwork` object comparing networks before vs after reduction.

`stability_plot` `ggplot` / `patchwork` object showing item stability before vs after reduction.

`overall` Named list with aggregated results across all item types. Under the default this contains:

`final_items` `data.frame` of final items across all types (columns as above).

`embeddings` Embeddings for the full reduced item set (see 'Notes on `embeddings`' below). Note: `overall$embeddings` does **not** include `selected`.

When `keep.org = TRUE` (in addition to defaults above): The top-level shape remains (`item_type_level` and `overall`) but includes original (pre-reduction) information:

`item_type_level` Each per-type sublist contains: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `initial_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, `stability_plot`.

`overall` Contains `final_items`, `initial_items`, and `embeddings` for the full item pool.

For `keep.org = TRUE`, per-type `embeddings` contains at least: `full_org`, `sparse_org`, `selected`, `full`, and `sparse`. (`overall$embeddings` contains the same subcomponents **except** `selected` is omitted.)

When `run.overall = TRUE` (`items.only = FALSE`, `embeddings.only = FALSE`):

`item_type_level` Same per-type structure as the default (see above).

`overall` A named list with aggregated results (not limited to `final_items` and `embeddings`) containing: `final_NMI`, `initial_NMI`, `embeddings`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, and `network_plot`.

When `all.together = TRUE` (regardless of `run.overall`): Results are **not** split into `item_type_level` and `overall`. Instead the function returns a single named list (applies to the full — possibly `keep.org` modified — result set) containing: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, and `stability_plot`.

References

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLOS ONE, 12*(6), e0174035. https://doi.org/10.1371/journal.pone.0174035

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research, 58*(6), 1165–1182. https://doi.org/10.1080/00273171.2023.2194606

Christensen, A. P., & Golino, H. (2021). Estimating the stability of psychological dimensions via bootstrap exploratory graph analysis: A Monte Carlo simulation and tutorial. *Psych, 3*(3), 479–500. https://doi.org/10.3390/psych3030032

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment, 2005*(9), P09008. https://doi.org/10.1088/1742-5468/2005/09/P09008

Russell-Lasalandra, L. L., Christensen, A. P., & Golino, H. (2024). Generative psychometrics via AI-GENIE: Automatic item generation and validation via network-integrated evaluation. https://osf.io/preprints/psyarxiv/fgbj4_v2.

Examples

```
## Not run:
##########################################################
#### Running AIGENIE with a downloaded LLM model ######
##########################################################

# Item type definitions
trait.definitions <- list(
 neuroticism = "Neuroticism is a personality trait that describes one's tendency to experience negative e
 extraversion = "Extraversion is a personality trait that describes people who are more focused on the ex
)

# Item attributes
aspects.of.personality.traits <- list(
 neuroticism = c("anxious", "depressed", "insecure", "emotional"),
 extraversion = c("friendly", "positive", "assertive", "energetic")
)

# Name the field or specialty
domain <- "Personality Measurement"

# Name the Inventory being created
scale.title <- "Two of 'Big Five:' A Streamlined Personality Inventory"

# Add a file path name to a local text generation model downloaded on your computer
model.path <- "ADD FILE PATH TO DOWNLOADED MODEL HERE"


# Generate and validate items using a model installed on your machine
local_example <- local_AIGENIE(
 item.attributes = aspects.of.personality.traits,
 item.type.definitions = trait.definitions,
 domain = domain,
 model.path = model.path
)


## End(Not run)
```

---

local_chat *Chat with a local LLM (no API calls)*

Description

Send one or more prompts to a locally available large-language model (LLM) without making remote API calls. The local model must be installed/available on the machine as a local model directory. This function is intended for fully local inference (no API key required).

Usage

```
local_chat(
  prompts,
  model.path,
  n.ctx = 4096,
  n.gpu.layers = -1,
  max.tokens = 1024,
  system.role = NULL,
  reps = 1,
  temperature = 1,
  top.p = 1,
  silently = FALSE
)
```

Arguments

`prompts`	A character string or character vector. The main prompt(s) given to the model. If multiple prompts are supplied, each will be sent separately to the model.
`model.path`	A character string. Path for the local model file. The function does not download models; ensure the model is present locally before using this function.
`n.ctx`	Integer, default `4096`. The context window (number of tokens) available to the model for a single generation.
`n.gpu.layers`	Integer, default `-1`. Number of model layers to place on GPU (if supported). Use `-1` to let the runtime choose automatically.
`max.tokens`	Integer, default `1024L`. Maximum number of tokens requested from the local model for a single generation.
`system.role`	A character string or character vector, default `NULL`. The system role(s) (model persona). If only one system role is provided and multiple prompts are supplied, the same role will be used for each prompt. If multiple system roles are provided, they should align with the prompts.
`reps`	Integer, default `1`. The number of times each prompt will be given to the model (independent generations).
`temperature`	Numeric, default `1`. Sampling temperature controlling response randomness.
`top.p`	Numeric, default `1`. Top-p (nucleus) sampling parameter.
`silently`	Logical, default `FALSE`. If `FALSE`, progress messages are printed to the console. If `TRUE`, the function runs quietly.

Details

Before running this function check that you are able to run the local environment via `check_local_llm_setup()`.

For each prompt × repetition the function constructs a `full_prompt` that includes the `system.role` and the user prompt, sets a deterministic generation seed per generation, and

calls the local model. A retry loop (up to 5 attempts, with brief waits) handles transient failures; if all attempts fail the function aborts with an informative error.

Value

A `data.frame` with one row per generation (i.e., per prompt × repetition) containing:

- `rep` — repetition index
- `prompt` — the prompt text sent to the model
- `response` — the raw text response returned by the model

Important / Warnings

- Local model inference can be resource intensive. Large models may require substantial disk space, RAM, and (optionally) GPU support. Performance and feasibility depend on model size and hardware.
- `model.path` must point to a model already present; this function will not download remote models.

See Also

chat

Examples

```
## Not run:
##################################################
### Example 1: Writing a Very Basic Prompt  #####
##################################################

# For local_chat you do NOT need an API key, but you DO need a text generation
# model available locally.
model <- "path/to/local-model" # replace with your local model path

# Then, write your prompt. This will be given to the model directly.
prompt <- "Why does the planet Saturn have rings? Give a 100 word explanation."

# Optionally, add a system role (a model persona)
system.role <- "You specialize in tutoring astronomy for high school students."

# Add the number of prompt repetitions. By default, this is set to 1. But it
# may be useful to increase the number of repetitions to get a sense of how
# consistent your output might be.
reps <- 3

# Now you are ready to chat with the local model
first_chat <- local_chat(
  model.path = model, # local model identifier or path
  prompts = prompt,
  system.role = system.role,
  reps = reps
```

```
)

# Check how the output changes from iteration to iteration
first_chat$response[[1]] # first iteration output
first_chat$response[[2]] # second iteration output
first_chat$response[[3]] # third iteration output


######################################################################
### Example 2: Send multiple prompts in a single function call #####
######################################################################

# You are able to send more than one prompt in a single call
prompt1 <- "Why does the planet Saturn have rings? Give a 100 word explanation."
prompt2 <- "Which planet is the hottest in our solar system? How do we know?"

# Aggregate the prompts in a single object
prompts <- c(prompt1, prompt2)

# Ask the model the questions
second_chat <- local_chat(
  model.path = model, # defined above
  prompts = prompts, # NEW
  system.role = system.role, # defined above
  reps = reps # defined above
)

# The outputted data frame for this example will have 6 rows
# since the number of prompts (2) times the number of reps (3)
# gives a total of 6 generations.
second_chat$response[second_chat$prompt == prompt1] # the responses from prompt 1
second_chat$response[second_chat$prompt == prompt2] # the responses from prompt 2


######################################################################
### Example 3: Send multiple prompts with different System Roles ###
######################################################################

# Perhaps your prompts are not related. In that case, you would probably want
# to set a different system role for each prompt.
prompt2 <- "What is the difference between eukaryotes and prokaryotes? Why?"

# This new second prompt does not fit with the astronomy tutor persona. Let's
# write a persona to match this new prompt topic.
system.role2 <- "You specialize in tutoring biology for middle school students."

# Now, let's combine the system roles into a single object
system.role <- c(system.role, # defined earlier: the astronomy tutor
                 system.role2 # defined above: the biology tutor
                 )

# Aggregate our prompts in a single object again
prompts <- c(prompt1, # Asks about Saturn's rings (needs astronomy tutor)
```

```
              prompt2 # Asks about types of cells (needs biology tutor)
              )

# Ask the model the questions
third_chat <- local_chat(
  model.path = model,
  prompts = prompts,
  system.role = system.role,
  reps = reps
)

# View the outputted data frame to examine the responses
View(third_chat)

## End(Not run)
```

`local_GENIE` *Local Generative Network-Integrated Evaluation (local_GENIE)*

Description

Local version of GENIE that uses locally installed embedding models for complete privacy and offline operation. Provides the same psychometric validation and quality assessment for user-supplied items as GENIE, but generates embeddings locally using transformer models instead of API calls.

Usage

```
local_GENIE(
  items,
  embedding.matrix = NULL,
  embedding.model = "bert-base-uncased",
  device = "auto",
  batch.size = 32,
  pooling.strategy = "mean",
  max.length = 512,
  EGA.model = NULL,
  EGA.algorithm = NULL,
  EGA.uni.method = NULL,
  uva.cut.off = 0.2,
  embeddings.only = FALSE,
  run.overall = FALSE,
  all.together = FALSE,
  plot = TRUE,
  silently = FALSE
)
```

Arguments

`items` Data frame with columns: statement, attribute, type, ID. All columns must be character type except ID (numeric or character allowed).

- `statement`: The actual item text
- `attribute`: The construct/attribute the item measures
- `type`: The item type/category
- `ID`: Unique identifier for each item

`embedding.matrix`
Optional numeric matrix or data frame where:

- Rows represent embedding dimensions
- Columns represent items (must match items$ID exactly)
- If `NULL`, embeddings will be generated using `embedding.model`

`embedding.model`
Local embedding model identifier or path. Compatible models:

- BERT variants: "bert-base-uncased", "bert-large-uncased"
- RoBERTa: "roberta-base", "roberta-large"
- DeBERTa: "microsoft/deberta-v3-base", "microsoft/deberta-v3-large"
- DistilBERT: "distilbert-base-uncased"
- Local paths: e.g., "/path/to/local/model"

`device` Device for embedding computation:

- "auto": Automatically detect best available device
- "cpu": Force CPU usage
- "cuda": Use NVIDIA GPU (if available)
- "mps": Use Apple Silicon GPU (if available)

`batch.size` Number of items to process simultaneously (default: 32)

`pooling.strategy`
Method for pooling token embeddings:

- "mean": Average all token embeddings (default)
- "cls": Use only the CLS token embedding
- "max": Max pooling across tokens

`max.length` Maximum sequence length for tokenization (default: 512)

`EGA.model` Network estimation model ("glasso", "TMFG", or NULL for auto-selection)

`EGA.algorithm` Community detection algorithm ("walktrap", "leiden", "louvain")

`EGA.uni.method`
Unidimensionality assessment method ("louvain", "expand", "LE")

`uva.cut.off` Numeric in `[0, 1)`. wTO threshold passed to `EGAnet::UVA` for the redundancy-reduction step (default: 0.20). Lower values remove more items.

`embeddings.only`
If `TRUE`, return embeddings and stop (skip network analysis)

`run.overall` A logical value (optional, default: FALSE). Controls whether a *fit* analysis on the complete item pool is run *post-reduction.* By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, an additional analysis is run on the overall sample, but no further reductions at the overall level are made. If only one item type is present, this argument will be ignored.

`all.together` A logical value (optional, default: FALSE). Controls whether the *reduction* analysis on the complete item pool is run. By default, only type-level reduction analyses are run (i.e., items of like-type go through the pipeline independent of the other items in the pool). When this flag is `TRUE`, reductions are made at the overall level (i.e., all items go through the reduction pipeline together, agnostic of item type). If only one item type is present, this argument will be ignored.

`plot` If `TRUE`, display network comparison plots

`silently` If `TRUE`, suppress progress messages

Value

Defaults: `items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `all.together = FALSE`.

When `items.only = TRUE`**:** Returns a `data.frame` of generated items with columns: `ID`, `statement`, `type`, and `attribute`.

When `embeddings.only = TRUE`**:** Returns a named `list` with two elements:

- `embeddings` — an embedding matrix/list (columns or rownames correspond to item IDs).
- `items` — the items `data.frame` described above.

Default behaviour (`items.only = FALSE`, `embeddings.only = FALSE`, `run.overall = FALSE`, `all.together = FALSE`): Returns a named `list` with two top-level elements:

`item_type_level` A named list where each name is an item type and each element is a per-type named list containing:

`final_NMI` Numeric: final normalized mutual information after reduction.

`initial_NMI` Numeric: initial NMI of the pre-reduced item pool.

`embeddings` List or matrix of embeddings for this item type (see 'Notes on `embeddings`' below).

`UVA` List from Unique Variable Analysis (contains at least `n_removed`, `n_sweeps`, `redundant_pairs` data.frame).

`bootEGA` List with bootEGA results (e.g. `initial_boot`, `final_boot`, `n_removed`, `items_removed`, `initial_boot_with_redundancies`).

`EGA.model_selected` Character: chosen EGA model (e.g. `"TMFG"` or `"Glasso"`).

`final_items` `data.frame`: final items after reduction (columns include `ID`, `statement`, `attribute`, `type`, `EGA_com`).

`final_EGA` EGA object (from EGAnet) after reduction.

`initial_EGA` Initial EGA object computed on the pre-reduced item set.

`start_N` Integer: initial number of items in this type.

`final_N` Integer: final number of items in this type.

`network_plot` `ggplot` / `patchwork` object comparing networks before vs after reduction.

`stability_plot` `ggplot` / `patchwork` object showing item stability before vs after reduction.

`overall` Named list with aggregated results across all item types. Under the default this contains:

`final_items` `data.frame` of final items across all types (columns as above).

`embeddings` Embeddings for the full reduced item set (see 'Notes on `embeddings`' below). Note: `overall$embeddings` does **not** include `selected`.

When `run.overall = TRUE` (`items.only = FALSE`, `embeddings.only = FALSE`):

`item_type_level` Same per-type structure as the default (see above).

`overall` A named list with aggregated results (not limited to `final_items` and `embeddings`) containing: `final_NMI`, `initial_NMI`, `embeddings`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, and `network_plot`.

When `all.together = TRUE` (regardless of `run.overall`): Results are **not** split into `item_type_level` and `overall`. Instead the function returns a single named list containing: `final_NMI`, `initial_NMI`, `embeddings`, `UVA`, `bootEGA`, `EGA.model_selected`, `final_items`, `final_EGA`, `initial_EGA`, `start_N`, `final_N`, `network_plot`, and `stability_plot`.

References

Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLOS ONE, 12*(6), e0174035. https://doi.org/10.1371/journal.pone.0174035

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research, 58*(6), 1165–1182. https://doi.org/10.1080/00273171.2023.2194606

Christensen, A. P., & Golino, H. (2021). Estimating the stability of psychological dimensions via bootstrap exploratory graph analysis: A Monte Carlo simulation and tutorial. *Psych, 3*(3), 479–500. https://doi.org/10.3390/psych3030032

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment, 2005*(9), P09008. https://doi.org/10.1088/1742-5468/2005/09/P09008

Russell-Lasalandra, L. L., Christensen, A. P., & Golino, H. (2024). Generative psychometrics via AI-GENIE: Automatic item generation and validation via network-integrated evaluation. https://osf.io/preprints/psyarxiv/fgbj4_v2.

Examples

```
## Not run:
#####################################################
#### Using GENIE with a Local Embedding Model  ####
#####################################################
```

```
# First, ensure that your machine is configured to compute local generation
install_local_llm_support()
# Once ready, continue to run GENIE on your data frame


# Specify item statements that you already have written
statements <- c(
  "I find myself naturally initiating conversations with strangers at social gatherings.",
  "I enjoy creating a welcoming atmosphere for people I meet for the first time.",
  "I generally maintain a hopeful outlook, even when faced with challenges.",
  "I frequently find myself in a good mood, spreading cheer to those around me.",
  "I often have the drive to engage in exciting activities, even after a long day.",
  "I tend to tackle projects with enthusiasm and high energy from start to finish.",
  "I actively seek to include others in group activities, making them feel part of the team.",
  "I frequently reach out to new acquaintances to foster connections and friendships.",
  "I habitually focus on the silver lining in difficult situations, maintaining an optimistic perspective
  "I often express gratitude for the positive aspects of my life, which enhances my overall mood.",
  "I find joy in taking on new challenges that require a burst of energy and enthusiasm.",
  "I thrive in dynamic environments that keep me on my toes and invigorate my spirit.",
  "I take pleasure in introducing people to one another, acting as a social connector.",
  "I enjoy making others comfortable by engaging them in light-hearted conversation.",
  "I often set a positive tone in group settings with my upbeat demeanor.",
  "I approach each day with a sense of excitement and a positive mindset.",
  "I am drawn to fast-paced environments where I can express my high energy levels.",
  "I feel invigorated when working on multiple projects that demand my full attention.",
  "I take delight in meeting new people and quickly making them feel at ease.",
  "I find it rewarding to help shy or reserved individuals become involved in group discussions.",
  "I have a natural tendency to uplift others with my positive remarks and outlook.",
  "I find happiness in highlighting the successes of others, contributing to a cheerful environment.",
  "I eagerly immerse myself in activities that demand stamina and sustained energy.",
  "I often channel my vitality into hobbies and sports that require physical exertion.",
  "I feel rejuvenated when I bring people together to collaborate and share ideas.",
  "I often extend a genuine greeting to others, creating an inviting atmosphere.",
  "I regularly see challenges as opportunities for growth and learning.",
  "I commonly radiate positivity, influencing the mood of those around me.",
  "I approach mornings with anticipation and vigor, ready to embrace the day.",
  "I consistently infuse enthusiasm into group activities, boosting collective energy levels.",
  "I make an effort to connect with people by remembering details about their lives.",
  "I genuinely enjoy learning about people's diverse experiences and viewpoints.",
  "I have a habit of encouraging others to see the bright side of their situations.",
  "I believe in celebrating small victories, finding joy in daily accomplishments.",
  "I often find myself eager to start the day with ambitious plans and goals.",
  "I am known for sustaining high levels of energy during extended work sessions or projects.",
  "I make an effort to engage those around me in meaningful and enjoyable conversations.",
  "I often seek opportunities to bring people together, fostering a sense of community.",
  "I naturally inspire others with my optimistic outlook, even in uncertain times.",
  "I frequently look for the positive aspects in challenging situations and share them with others.",
  "I approach new experiences with an eagerness and fervor that motivates those around me.",
  "I thrive on maintaining high energy levels throughout demanding and fast-paced days.",
  "I take pleasure in initiating warm interactions in group settings to make everyone comfortable.",
  "I enjoy hosting gatherings that connect friends and encourage social bonding.",
  "I am skilled at turning setbacks into learning experiences to maintain a positive outlook.",
```

```
"I always try to highlight the benefits in situations, enhancing a cheerful atmosphere.",
"I find excitement in starting the day with a list of activities to energize my routine.",
"I relish the challenge of keeping up with dynamic schedules that require sustained energy.",
"I often find joy in making newcomers feel welcome and appreciated in group settings.",
"I genuinely enjoy striking up conversations to learn more about the people I encounter.",
"I have a knack for seeing potential in situations that others might overlook.",
"I consistently try to uplift the mood in my surroundings with hopeful and encouraging words.",
"I frequently harness my energy to inspire and motivate those around me in team environments.",
"I often feel invigorated by challenges that require sustained focus and dynamic thinking.",
"I often create environments where people feel encouraged to share their thoughts freely.",
"I find it fulfilling to engage deeply with people, building lasting connections.",
"I see potential in every day, believing it holds opportunities for something good.",
"I actively focus on the pleasures of life, which naturally enhances my mood.",
"I am invigorated by opportunities to engage in lively and spirited events.",
"I tend to maintain momentum throughout the day, sustaining my energy levels.",
"I frequently experience sudden shifts in my emotions even when there is no apparent reason.",
"People often find it difficult to predict my emotional reactions to different situations.",
"I often doubt my abilities and worry about whether I am meeting expectations.",
"I frequently question my self-worth and tend to seek reassurance from others.",
"I become annoyed easily over small inconveniences or delays.",
"I often find myself feeling agitated or frustrated in situations that don't bother most people.",
"My mood can change drastically over the course of a day, often without any clear reason.",
"I tend to experience emotional highs and lows more intensely than those around me.",
"I sometimes avoid taking on new challenges because I fear not being good enough.",
"I often feel uncertain about my social standing and worry about being accepted by others.",
"I find myself getting irritated quickly when things don't go my way.",
"Minor annoyances often cause my patience to wear thin unusually fast.",
"I frequently struggle to maintain a stable emotional state throughout the day.",
"Unexpected events can cause me to experience drastic emotional swings.",
"I often feel inadequate in comparison to others around me.",
"I tend to second-guess my choices due to a lack of confidence in myself.",
"I tend to become frustrated when things do not proceed as I have planned.",
"I am prone to irritation when faced with unexpected changes to my routine.",
"My emotional state is often unpredictable, shifting from contentment to sadness with little warning.",
"People have commented that my emotions seem to fluctuate more than those of others.",
"I frequently feel self-conscious about my achievements compared to those of my peers.",
"I often worry excessively about making mistakes, even in situations where it might be inconsequential.
"Small disruptions in my daily routine can trigger strong feelings of annoyance.",
"I find myself becoming irritated more quickly than others when under stress or pressure.",
"I often find my emotional responses to be unpredictable, feeling fine one moment and unsettled the nex
"I experience strong emotions that can shift unexpectedly, often catching me off guard.",
"I regularly feel uncertain about my ability to manage new responsibilities effectively.",
"I often question my decisions, fearing they might not lead to the best outcomes.",
"I frequently find myself reacting with impatience to situations perceived as minor interruptions.",
"Even minor provocations can sometimes lead to an exaggerated sense of annoyance for me.",
"My emotional state is often inconsistent, and I can feel ecstatic or despondent within short timeframe
"I notice that my feelings can be quite volatile and intense, affecting how I interact with others thro
"I regularly doubt whether I am capable of achieving my personal or professional goals.",
"I often seek validation from others to feel reassured about my self-worth.",
"I am sensitive to disturbances and find my patience wearing thin quickly when things aren't orderly.",
"I occasionally struggle to contain my annoyance over trivial issues that disrupt my sense of calm.",
"I can go from feeling upbeat to being downcast without an obvious cause.",
"My emotional responses can sometimes be unpredictable, shifting with little notice.",
```

```
  "I often feel the need for affirmation about my abilities from friends or colleagues.",
  "I tend to compare myself to others and feel uncertain about my achievements.",
  "I find myself easily bothered by noises or disturbances in my environment.",
  "I get easily flustered by situations that interrupt my planned activities.",
  "I find it challenging to maintain a consistent emotional state, regardless of external situations.",
  "My emotional reactions can be intense and differ significantly from moment to moment.",
  "I have a persistent fear of not measuring up to the expectations placed on me.",
  "I often feel anxious about others' perceptions of my capabilities and appearance.",
  "I am quick to express frustration at minor inconveniences in my daily routine.",
  "I find that small, unforeseen events often disrupt my sense of calm and lead to irritation.",
  "My emotional reactions can be strong and relentless, impacting my behavior throughout the day.",
  "I often find myself emotionally labile, with an inner turbulence that others rarely perceive.",
  "I frequently worry about my competence in areas where others seem confident.",
  "I have a tendency to second-guess myself and require affirmation to feel reassured about my choices.",
  "Small disruptions can ignite a lingering sense of agitation within me.",
  "I often catch myself feeling irritable even in relatively calm settings.",
  "I find myself swinging from happy to melancholic in a short span of time, often surprising even myself
  "Others often comment on how quickly my mood can change in response to seemingly minor events.",
  "I tend to feel apprehensive about presenting my opinions, fearing they may be judged harshly.",
  "I often require reassurance from peers to feel confident in my decisions and ideas.",
  "Interruptions during focused tasks often lead to an outpour of irritation from me.",
  "I struggle to keep my frustration in check when things do not unfold as expected."
)


# Create the item type and attribute labels
item.attributes <- c(
  rep(c("friendly", "positive", "energetic"), each = 2, times = 10),
  rep(c("moody", "insecure", "irritable"), each = 2, times = 10)
)
item.types <- c(
  rep("extraversion", 60),
  rep("neuroticism", 60)
)


# Build your data frame with the required columns: ID, statement, attribute, and type
items_df <- data.frame(
  ID = rep(as.factor(1:length(statements))),
  statement = statements,
  attribute = item.attributes,
  type = item.types
)


# Run GENIE with items you provide with the default local embedding model ("bert-base-uncased")
example_reduction <- local_GENIE(items = items_df)

# View the results
View(example_reduction)


##############################################################################################
```

```
###### Or, Run local_GENIE with a locally-install Embedding Model of your Choice #######
################################################################################################

# Provide the path to a compatible locally installed embedding model
# Note that this package will not install the model for you, it should already be
# installed and ready to go
embedding.model <- "ADD YOUR PATH HERE"

# Run GENIE using the Hugging Face Embedding model
example_reduction_your_model <- local_GENIE(items = items_df,
                                embedding.model = embedding.model)


## End(Not run)
```

`main.prompts_validate`

Validate and Normalize `main.prompts`

Description

Validates that `main.prompts` is a named list of non-empty strings, one for each attribute in `items.attributes`, matched by normalized name.

Usage

```
main.prompts_validate(main.prompts, items.attributes, silently)
```

Arguments

`main.prompts`	A named list of prompt strings, one per attribute.
`items.attributes`	A cleaned list from `validate_items.attributes()`.
`silently`	A flag determining wheter a warning message should be printed

Value

A cleaned and ordered named list of trimmed prompt strings. Also returns the appropriate 'custom' flag (TRUE if custom ok, FALSE if not)

max.tokens_validate *Check that max.tokens is an integer*

Description

Check that max.tokens is an integer

Usage

```
## S3 method for class 'tokens_validate'
max(max.tokens, silently)
```

Arguments

`max.tokens`	The number of tokens requested for the text generator
`silently`	Whether to print the warning statements

Value

if valid, the `max.tokens` value as an integer object type

plot_comparison *Plot Comparisons*

Description

Generates a comparative plot of two network analysis results, typically representing the item network before and after AI-GENIE reduction. The plot includes provided captions, displays NMI values for each network, and incorporates a scale title to contextualize the comparison. The layout may be adjusted based on the `ident` parameter.

Usage

```
plot_comparison(p1, p2, caption1, caption2, nmi2, nmi1, title)
```

Arguments

`p1`	An object representing the first network analysis result (e.g., the initial EGA object before reduction).
`p2`	An object representing the second network analysis result (e.g., the final EGA object after reduction).
`caption1`	A character string to be used as a caption or title for the first network (e.g., "Before AI-GENIE Network").
`caption2`	A character string for the second network (e.g., "After AI-GENIE Network").

`nmi2` A numeric value representing the NMI of the second network.

`nmi1` A numeric value representing the Normalized Mutual Information (NMI) of the first network.

`title` A character string specifying the title of the plot.

Value

A plot object that visually compares the two network structures. The plot will typically display the two networks (either side-by-side or in an overlaid manner) with the provided captions and NMI values. The exact type of the plot object (e.g., a `ggplot` object or a base R plot) depends on the implementation.

`plot_stability_comparison` *Plot Stability Comparison (network + item stability dotplot, side by side)*

Description

Builds a 4-panel comparison: pre-reduction network + pre-reduction item stability, next to post-reduction network + post-reduction item stability. Mirrors the layout of the AIGENIE simulation/reference figure.

Usage

```
plot_stability_comparison(boot1, boot2, caption1, caption2, nmi1, nmi2, title)
```

Arguments

`boot1`, `boot2` bootEGA objects (pre and post reduction).

`caption1`, `caption2` Captions under each network panel.

`nmi1`, `nmi2` NMI values pre/post.

`title` Overall title.

Value

A patchwork object combining the four panels.

print_results *Print Results*

Description

Displays a summary of the AI-GENIE analysis results, including the EGA model used, embedding type, starting and final number of items, and NMI values before and after reduction. The summary includes the number of iterations for both UVA (Unique Variable Analysis) and bootstrapped EGA steps.

Usage

```
print_results(obj, obj2, run.overall)
```

Arguments

`obj`	A list object containing the OVERALL analysis results returned by `get_results`.
`obj2`	A list object containing the ITEM-TYPE LEVEL analysis results returned by `get_results`.
`run.overall`	A flag denoting if overall results should be printed

Value

No return value; the function prints the results to the console.

python_env_info *Get AI-GENIE Python Environment Info*

Description

Returns diagnostic information about the AIGENIE Python environment, including paths, installation status, and installed packages. Useful for troubleshooting Python-related issues.

Usage

```
python_env_info()
```

Value

A list with the following elements:

`env_path`	Path to the virtual environment directory.
`python_path`	Path to the Python executable.
`env_exists`	Logical. Whether the environment directory exists.
`python_exists`	Logical. Whether the Python executable exists.

`initialized` Logical. Whether Python has been initialized this session.

`uv_available` Logical. Whether UV is installed and accessible.

`installed_packages`
Character vector of installed packages (if environment exists).

See Also

`reinstall_python_env` to fix environment issues.

Examples

```
## Not run:
# Check environment status
info <- python_env_info()

# Is the environment set up?
info$env_exists

# What packages are installed?
cat(info$installed_packages, sep = "\n")

# Is UV available?
info$uv_available

## End(Not run)
```

`reduce_redundancy_uva`
Reduce Redundancy via Iterative UVA (with Redundant Pair Logging)

Description

Applies EGAnet::UVA iteratively and logs human-readable redundant item sets.

Usage

```
reduce_redundancy_uva(
  embedding_matrix,
  items,
  corr = "auto",
  uva.cut.off = 0.2
)
```

Arguments

`embedding_matrix`	A numeric matrix of embeddings (columns = items).
`items`	Data frame with `ID` and `statement` columns.
`corr`	Character. Correlation method to use. Default "auto" uses EGAnet's automatic correlation detection. Other options: "pearson", "spearman", "cosine".
`uva.cut.off`	Numeric in `[0, 1)`. The weighted topological overlap threshold passed to `EGAnet::UVA`. Items with pairwise wTO at or above this value are flagged as redundant. Default `0.20`.

Value

A list with the reduced matrix, sweep metadata, and redundancy log.

`reinstall_python_env` *Reinstall AI-GENIE Python Environment*

Description

Removes and recreates the Python virtual environment with all required dependencies. Use this function if you encounter Python-related errors, want to update Python packages, or need to change the environment configuration.

AIGENIE uses UV (https://docs.astral.sh/uv/) for fast, reliable Python environment management.

Usage

```
reinstall_python_env(
  include_huggingface = TRUE,
  include_local_llm = FALSE,
  gpu = FALSE
)
```

Arguments

`include_huggingface`	Logical. Include HuggingFace packages (transformers, sentence-transformers, torch). Required for local embeddings with HuggingFace models. Default `TRUE`.
`include_local_llm`	Logical. Include llama-cpp-python for running local GGUF models with `local_AIGENIE`. Default `FALSE`.
`gpu`	Logical. Install GPU-enabled PyTorch. Requires CUDA-compatible NVIDIA GPU and proper driver installation. Default `FALSE`.

Value

Invisible `TRUE` on success.

See Also

`python_env_info` to check environment status, `install_gpu_support` for GPU setup, `install_local_llm_support` for local model setup.

Examples

```
## Not run:
# Fix Python environment issues
reinstall_python_env()

# Reinstall with GPU support
reinstall_python_env(gpu = TRUE)

# Minimal install (API-only, no HuggingFace - faster)
reinstall_python_env(include_huggingface = FALSE)

# Full install with local LLM support
reinstall_python_env(include_huggingface = TRUE, include_local_llm = TRUE)

## End(Not run)
```

`resolve_model_name` *Resolve and Normalize Model Name*

Description

Accepts a free-form model name and returns a standardized string. Known aliases are resolved to canonical model names. If the model is not recognized, a warning is issued and the cleaned original input is returned.

Usage

```
resolve_model_name(model, silently)
```

Arguments

`model`	A single string, the user-supplied model name.
`silently`	A flag to determine if warnings should be printed to the screen.

Value

A standardized model name.

`response.options_validate` *Validate and Clean* `response.options`

Description

Validates that `response.options` is an atomic vector of non-empty strings, with no missing or invalid values. Whitespace is trimmed from each string.

Usage

```
response.options_validate(response.options)
```

Arguments

`response.options`	An atomic character vector of response labels.

`run_all_together` *Modify the items data frame to run the reduction on all items together*

Description

Modify the items data frame to run the reduction on all items together

Usage

```
run_all_together(items)
```

Arguments

`items`	A data frame containing the items either generated by AI or supplied by the user

Value

A data frame whose ”attribute” and ”type” columns have been modifies so that the entire sample runs together

run_flags_validate *Check that the* run.overall *and* all.together *flags are logically consistent with the number of item types.*

Description

Check that the run.overall and all.together flags are logically consistent with the number of item types.

Usage

```
run_flags_validate(run.overall, all.together, item.attributes, silently)
```

Arguments

run.overall	If a final quality analysis should be run on the overall sample
all.together	If the reduction analysis should be run on all of the items agnostic of item type
item.attributes	A named list of attributes and item types.
silently	whether the print statements should appear

Value

a named list with the updated all.together and run.overall flags

run_item_reduction_pipeline *Run reduction pipeline for all item types*

Description

Run reduction pipeline for all item types

Usage

```
run_item_reduction_pipeline(
  embedding_matrix,
  items,
  EGA.model = NULL,
  EGA.algorithm = "walktrap",
  EGA.uni.method = "louvain",
  corr = "auto",
  ncores = NULL,
  boot.iter = 100,
```

```
  uva.cut.off = 0.2,
  keep.org,
  silently,
  plot
)
```

Arguments

`embedding_matrix`	Full embedding matrix (columns = all items)
`items`	Data frame of all items (must include ID, statement, attribute, type)
`EGA.model`	NULL, "glasso", or "TMFG"
`EGA.algorithm`	EGA algorithm
`EGA.uni.method`	EGA uni.method
`corr`	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.
`ncores`	Numeric. Number of cores for parallel processing.
`boot.iter`	Numeric. Number of bootstrap iterations.
`uva.cut.off`	Numeric in `[0, 1)`. wTO threshold for `EGAnet::UVA`. Default `0.20`.
`keep.org`	Logical. Whether to include original items and embeddings
`silently`	Logical. Whether to print progress statements
`plot`	Logical. Whether to plot the network plots at the end

Value

A named list of pipeline results, one per item type

`run_pipeline_for_all` *Run full pipeline for all items in the sample*

Description

Run full pipeline for all items in the sample

Usage

```
run_pipeline_for_all(
  item_level,
  items,
  embeddings,
  model = NULL,
  algorithm = "walktrap",
  uni.method = "louvain",
  corr = "auto",
```

```
  ncores = NULL,
  boot.iter = 100,
  uva.cut.off = 0.2,
  keep.org = FALSE,
  silently,
  plot
)
```

Arguments

item_level	AIGENIE results on the item level
items	all items generated for the initial item pool
embeddings	all embeddings created for the initial item pool
model	NULL, "glasso", or "TMFG"
algorithm	EGA algorithm
uni.method	EGA uni.method
corr	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.
ncores	Numeric. Number of cores for parallel processing.
boot.iter	Numeric. Number of bootstrap iterations.
uva.cut.off	Numeric in `[0, 1)`. wTO threshold for `EGAnet::UVA`. Default `0.20`.
keep.org	Logical. Whether to include original items and embeddings
silently	Logical. Whether to print progress statements
plot	logical. Whether to plot the network plot

Value

A named list containing pipeline results for this type

---

run_pipeline_for_item_type

Run full pipeline for a single item type

Description

Run full pipeline for a single item type

Usage

```
run_pipeline_for_item_type(
  embedding_matrix,
  items,
  type_name,
  model = NULL,
  algorithm = "walktrap",
  uni.method = "louvain",
  corr = "auto",
  ncores = NULL,
  boot.iter = 100,
  uva.cut.off = 0.2,
  keep.org = FALSE,
  silently,
  plot
)
```

Arguments

`embedding_matrix`	Numeric matrix (columns = items for one type)
`items`	Data frame of items for this type (must include ID, statement, attribute)
`type_name`	Character. Type label used for tracking/logging.
`model`	NULL, "glasso", or "TMFG"
`algorithm`	EGA algorithm
`uni.method`	EGA uni.method
`corr`	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.
`ncores`	Numeric. Number of cores for parallel processing. Default NULL uses EGAnet default.
`boot.iter`	Numeric. Number of bootstrap iterations. Default 100.
`uva.cut.off`	Numeric in `[0, 1)`. wTO threshold for `EGAnet::UVA`. Default `0.20`.
`keep.org`	Logical. Whether to include original items and embeddings
`silently`	Logical. Whether to print progress statements
`plot`	Logicial. Whether to plot the network plots at the end

Value

A named list containing pipeline results for this type

`select_optimal_embedding` *Select Optimal Embedding and EGA Model Based on NMI*

Description

Select Optimal Embedding and EGA Model Based on NMI

Usage

```
select_optimal_embedding(
  embedding_matrix,
  sparse_matrix,
  true_communities,
  model = NULL,
  algorithm = "walktrap",
  uni.method = "louvain",
  corr = "auto"
)
```

Arguments

`embedding_matrix`	A numeric matrix (columns = items). The full (dense) representation.
`sparse_matrix`	A numeric matrix (columns = items) giving the sparse representation, aligned to `embedding_matrix` (same items and column order). This is computed once on the pre-UVA pool and then subset to the post-UVA items in the AI-GENIE pipeline; passing it in (rather than recomputing inside) preserves the pre-UVA quantile thresholds.
`true_communities`	A named list of known communities.
`model`	Character. One of "glasso", "TMFG", or NULL (to test both).
`algorithm`	Community detection algorithm (e.g., "walktrap").
`uni.method`	Unidimensionality method (e.g., "louvain").
`corr`	Character. Correlation method. Default "auto" uses EGAnet's automatic detection.

Value

A list with best embedding, model, communities, NMI, and comparison log.

`set_huggingface_token` *Set Hugging Face Token*

Description

Configure your HuggingFace API token for accessing gated models like Google's EmbeddingGemma or other restricted models.

Usage

```
set_huggingface_token(token, save = TRUE)
```

Arguments

`token`	Character. Your HuggingFace API token from https://huggingface.co/settings/tokens.
`save`	Logical. If `TRUE` (default), saves the token to the HuggingFace cache for future sessions. If `FALSE`, sets it only for the current R session.

Details

Some embedding models on HuggingFace require authentication:

- `google/embeddinggemma-300m`
- Other gated models

Before using these models, you must:

1. Create an account at https://huggingface.co
2. Accept the model's license on its model page
3. Generate an access token at https://huggingface.co/settings/tokens
4. Call this function with your token

Value

Invisible `TRUE` on success.

Examples

```
## Not run:
# Set token (saved permanently for future sessions)
set_huggingface_token("hf_xxxxxxxxxxxxxxxx")

# Set token for this session only (not saved)
set_huggingface_token("hf_xxxxxxxxxxxxxxxx", save = FALSE)

# Now you can use gated HuggingFace models
```

```
results <- GENIE(
  items = my_items,
  embedding.model = "BAAI/bge-large-en-v1.5",
  hf.token = "hf_xxxxxxxxxxxxxxxxx"
)

## End(Not run)
```

`sparsify_embeddings` *Sparsify Embedding Matrix*

Description

Applies sparsification to an embedding matrix by zeroing out values between specified quantiles. Includes fallback strategies if initial sparsification results in all zeros.

Usage

```
sparsify_embeddings(
  embedding_matrix,
  lower_quantile = 0.025,
  upper_quantile = 0.975,
  fallback_lower = 0.1,
  fallback_upper = 0.9
)
```

Arguments

`embedding_matrix`
: Numeric matrix with items as columns, dimensions as rows

`lower_quantile`
: Lower quantile threshold (default 0.025)

`upper_quantile`
: Upper quantile threshold (default 0.975)

`fallback_lower`
: Fallback lower quantile if first attempt fails (default 0.10)

`fallback_upper`
: Fallback upper quantile if first attempt fails (default 0.90)

Details

Sparsification process:

1. Zero out values between lower and upper quantiles
2. If result is all zeros, try fallback quantiles
3. If still all zeros, return original matrix

`silently` is always `TRUE`. It is only set to `FALSE` for developement and diagnostic purposes.

Value

Sparsified embedding matrix with same dimensions as input

---

target.N_validate *Validate and Expand* target.N *for Each Item Attribute*

Description

Ensures that `target.N` is either:

- NULL -> defaults to 60 per attribute
- A single integer -> repeated for each attribute
- A list/vector of integers -> must match number of attributes

Usage

```
target.N_validate(
  target.N,
  items.attributes,
  items.only,
  embeddings.only,
  silently
)
```

Arguments

`target.N`	An integer, list/vector of integers, or NULL.
`items.attributes`	A cleaned list returned from `validate_items.attributes()`.
`items.only`	A flag used to determine if only items need to be generated
`embeddings.only`	A flag used to determine if only embeddings need to be generated
`silently`	A flag used to determine if warnings should be printed

Value

A list of integers, one per attribute (named).

temperature_validate *Validate* temperature *for Text Generation*

Description

Ensures temperature is a numeric value between 0 and 2

Usage

```
temperature_validate(temperature)
```

Arguments

temperature	A numeric value

top.p_validate *Validate* top.p *for Text Generation*

Description

Ensures top.p is a numeric value between 0 and 1, or NULL.

Usage

```
top.p_validate(top.p)
```

Arguments

top.p	A numeric value

uva.cut.off_validate *Validate* uva.cut.off

Description

Ensures uva.cut.off is a single numeric value in [0, 1).

Usage

```
uva.cut.off_validate(uva.cut.off)
```

Arguments

uva.cut.off	A numeric value.

validate_booleans *Validate Boolean Arguments*

Description

Validates that all arguments passed to the function are scalar boolean values (`TRUE` or `FALSE`). If any argument is not a boolean, an error is thrown that identifies the offending variable by name and instructs the user to set it to either `TRUE` or `FALSE`.

Usage

```
validate_booleans(...)
```

Arguments

`...`	One or more variables to check. We are expecting each to be a logical scalar (`TRUE` or `FALSE`). In `AIGENIE`, these variables would be `items.only`, `adaptive`, `plot`, `keep.org`, `silently`, and `embeddings.only`.

validate_ega_params *Validate EGA Parameters*

Description

Validates and normalizes the EGA algorithm, unidimensionality method, and model parameters. Trims whitespace and performs case-insensitive matching. Returns canonical-cased values.

Usage

```
validate_ega_params(EGA.algorithm, EGA.uni.method, EGA_model)
```

Arguments

`EGA.algorithm`	A string: one of ”leiden”, ”louvain”, ”walktrap” (or NULL, in which case default behavior takes over)
`EGA.uni.method`	A string: one of ”expand”, ”LE”, ”louvain”
`EGA_model`	A string or NULL: one of ”glasso”, ”TMFG”

Value

A named list with cleaned and correctly-cased values.

`validate_local_embedding_model` *Validate Local Embedding Model*

Description

Validates that the embedding model is appropriate for local raw embeddings. Checks for BERT-family models that provide raw feature extraction.

Usage

```
validate_local_embedding_model(embedding.model, silently = FALSE)
```

Arguments

`embedding.model`	Character string specifying model identifier or path
`silently`	Logical. Suppress informational messages

Value

The validated model identifier or path

`validate_local_embedding_params` *Validate Local Embedding Parameters*

Description

Validates parameters specific to local embedding generation

Usage

```
validate_local_embedding_params(
  device,
  batch.size,
  pooling.strategy,
  max.length
)
```

Arguments

`device`	Device for computation ("auto", "cpu", "cuda", "mps")
`batch.size`	Number of items to process simultaneously
`pooling.strategy`	Strategy for pooling token embeddings
`max.length`	Maximum sequence length for tokenization

Value

A list of validated parameters

`validate_local_llm_params` *Validate Local LLM Generation Parameters*

Description

Validates parameters specific to local LLM generation

Usage

```
validate_local_llm_params(n.ctx, n.gpu.layers, max.tokens)
```

Arguments

`n.ctx`	Context window size
`n.gpu.layers`	Number of layers to offload to GPU
`max.tokens`	Maximum tokens for generation

Value

A list of validated parameters

`validate_model.path` *Validate Local Model Path*

Description

Validate Local Model Path

Usage

```
validate_model.path(model.path, silently = FALSE)
```

Arguments

`model.path`	Path to local GGUF model file
`silently`	Logical. Suppress warnings

Value

The expanded, validated path

validate_prompt.notes *Validate and Normalize* prompt.notes

Description

Accepts a string, NULL, or a named list of strings/NULLs. Ensures one entry per attribute in `items.attributes`, returning a fully named and cleaned list.

Usage

```
validate_prompt.notes(prompt.notes, items.attributes)
```

Arguments

`prompt.notes`	A single string, NULL, or named list of strings/NULLs.
`items.attributes`	A cleaned list from `validate_items.attributes()`.

Value

A named list of strings, one per attribute, with NULLs replaced by "".

validate_reps *Check that reps is an integer*

Description

Check that reps is an integer

Usage

```
validate_reps(reps)
```

Arguments

`reps`	The number of repetitions per prompt requested

Value

if valid, the `reps` value as an integer object type

validate_strings *Validate That Inputs Are Strings*

Description

Ensures that each argument is a single, non-NA string. Throws an error if any argument is not a character scalar.

Usage

```
validate_strings(...)
```

Arguments

...	One or more variables to validate.

validate_system.role_prompts *Checks* system.role *and* prompts *for the* chat *function*

Description

Checks system.role and prompts for the chat function

Usage

```
validate_system.role_prompts(system.role, prompts)
```

Arguments

system.role	The persona for the model. Either a string, NULL (default), or a list of strings
prompts	The prompts to be given to the model. Either a string or a list of strings

Value

if valid, a list with the system.role and prompts objects

validate_user_input_AIGENIE
Validate All User Inputs for AI-GENIE

Description

This function performs comprehensive validation and normalization of all user-supplied inputs to the AI-GENIE package. It checks logical flags, strings, model names, item attribute structures, and ensures consistency across all interdependent components.

Usage

```
validate_user_input_AIGENIE(
  item.attributes,
  openai.API,
  hf.token,
  main.prompts,
  groq.API,
  anthropic.API,
  jina.API,
  model,
  temperature,
  top.p,
  embedding.model,
  target.N,
  domain,
  scale.title,
  item.examples,
  audience,
  item.type.definitions,
  response.options,
  prompt.notes,
  system.role,
  EGA.model,
  EGA.algorithm,
  EGA.uni.method,
  keep.org,
  items.only,
  embeddings.only,
  adaptive,
  run.overall,
  all.together,
  plot,
  silently
)
```

Arguments

`item.attributes`	A named list of attributes and item types. Must be validated via `item.attributes_validate()`.
`openai.API`	A string. OpenAI API key.
`hf.token`	A string. HuggingFace API key.
`main.prompts`	A named list of custom prompts that the user specifies (if desired)
`groq.API`	A string or NULL. Groq API key.
`model`	A string. The user-specified language model. Will be resolved to a canonical model name using `normalize_model_name()`.
`temperature`	A numeric value between 0 and 2.
`top.p`	A numeric value between 0 and 1.
`embedding.model`	A string or NULL. Must be one of the accepted OpenAI embedding models.
`target.N`	Either a scalar integer, NULL, or a named list/vector of integers corresponding to each attribute. Used for synthetic item generation.
`domain`	A string describing the domain of the assessment.
`scale.title`	A string naming the scale.
`item.examples`	A data frame containing `type`, `attribute`, and `statement` columns. All values must be strings. Optional.
`audience`	A string or NULL. The intended audience of the assessment.
`item.type.definitions`	A named list mapping item types to their descriptions. Optional.
`response.options`	An atomic vector of strings listing the response options users will have. Optional.
`prompt.notes`	A named list or string that gives the LLM additional instructions to be appended to the prompt. Optional.
`system.role`	A string or NULL. Used to customize the system prompt.
`EGA.model`	A string or NULL. One of `"BGGM"`, `"glasso"`, or `"TMFG"`.
`EGA.algorithm`	A string. One of `"leiden"`, `"louvain"`, or `"walktrap"`.
`EGA.uni.method`	A string. One of `"expand"`, `"LE"`, or `"louvain"`.
`keep.org`	A boolean. If TRUE, preserve original inputs in the output.
`items.only`	A boolean. Whether to generate only items.
`embeddings.only`	A boolean. Whether to run in embedding-only mode.
`adaptive`	A boolean. Whether adaptive design logic should be applied.
`plot`	A boolean. Whether to display plots for visual diagnostics.
`silently`	A boolean. If TRUE, suppresses warning messages.

Details

If any input is invalid or misaligned with the package's expected structure, informative errors or warnings are raised. Cleaned and normalized objects are returned for use downstream.

Value

A named list containing:

target.N A named list of integers, aligned with `item.attributes`
EGA.model Canonical model string or NULL
EGA.uni.method Canonical unidimensionality method
EGA.algorithm Canonical community detection algorithm
model Resolved model string for text generation
item.type.definitions Cleaned item type definitions (if provided)
item.examples Cleaned item examples (if provided)
item.attributes Cleaned and normalized item attributes
prompt.notes Cleaned and normalized prompt notes (if provided)
main.prompts Cleaned and normalized main prompts (if provided)
custom A flag signaling whether we are in custom mode or not

`validate_user_input_GENIE` *Validate All User Inputs for GENIE*

Description

Validate All User Inputs for GENIE

Usage

```
validate_user_input_GENIE(
  items,
  embedding.matrix,
  openai.API,
  hf.token,
  jina.API,
  embedding.model,
  EGA.model,
  EGA.algorithm,
  EGA.uni.method,
  embeddings.only,
  run.overall,
  all.together,
  plot,
  silently
)
```

Arguments

`items`	A data frame with columns: statement, attribute, type, ID
`embedding.matrix`	Optional numeric matrix/data frame with items as columns
`openai.API`	OpenAI API key (string or NULL)
`hf.token`	HuggingFace token (string or NULL)
`jina.API`	Jina API key (string or NULL)
`embedding.model`	Embedding model identifier (string)
`EGA.model`	EGA network model (string or NULL)
`EGA.algorithm`	EGA algorithm (string)
`EGA.uni.method`	EGA unidimensionality method (string)
`embeddings.only`	Whether to stop after embeddings (boolean)
`plot`	Whether to show plots (boolean)
`silently`	Whether to suppress messages (boolean)

Value

A named list containing all validated and normalized parameters

---

`validate_user_input_local_AIGENIE` *Validate All User Inputs for Local AI-GENIE*

Description

Comprehensive validation of all inputs for local model execution. Reuses existing validators where applicable and adds local-specific validations.

Usage

```
validate_user_input_local_AIGENIE(
  item.attributes,
  model.path,
  embedding.model,
  main.prompts,
  temperature,
  top.p,
  target.N,
  domain,
  scale.title,
  item.examples,
```

```
  audience,
  item.type.definitions,
  response.options,
  prompt.notes,
  system.role,
  EGA.model,
  EGA.algorithm,
  EGA.uni.method,
  n.ctx,
  n.gpu.layers,
  max.tokens,
  device,
  batch.size,
  pooling.strategy,
  max.length,
  keep.org,
  items.only,
  embeddings.only,
  adaptive,
  run.overall,
  all.together,
  plot,
  silently
)
```

Arguments

`item.attributes`	Named list of attributes (same as API version)
`model.path`	Path to local GGUF model
`embedding.model`	Local embedding model identifier
`main.prompts`	Optional custom prompts
`temperature`	LLM temperature
`top.p`	LLM top-p sampling
`target.N`	Target number of items
`domain`	Assessment domain
`scale.title`	Scale name
`item.examples`	Example items
`audience`	Target audience
`item.type.definitions`	Type definitions
`response.options`	Response scale options
`prompt.notes`	Additional prompt instructions

system.role	System prompt
EGA.model	EGA model type
EGA.algorithm	EGA algorithm
EGA.uni.method	EGA unidimensionality method
n.ctx	Context window size
n.gpu.layers	GPU layers
max.tokens	Maximum generation tokens
device	Embedding computation device
batch.size	Embedding batch size
pooling.strategy	Embedding pooling strategy
max.length	Embedding max sequence length
keep.org	Keep original data
items.only	Generate items only
embeddings.only	Generate embeddings only
adaptive	Use adaptive generation
plot	Show plots
silently	Suppress messages

Value

A list of all validated parameters

---

validate_user_input_local_GENIE

Validate All User Inputs for Local GENIE

Description

Validate All User Inputs for Local GENIE

Usage

```
validate_user_input_local_GENIE(
  items,
  embedding.matrix,
  embedding.model,
  device,
  batch.size,
  pooling.strategy,
  max.length,
```

```
  EGA.model,
  EGA.algorithm,
  EGA.uni.method,
  embeddings.only,
  run.overall,
  all.together,
  plot,
  silently
)
```

Arguments

`items`	Data frame with columns: statement, attribute, type, ID
`embedding.model`	Local embedding model identifier or path
`device`	Device for embeddings ("auto", "cpu", "cuda", "mps")
`batch.size`	Batch size for embedding generation
`pooling.strategy`	Pooling strategy ("mean", "cls", "max")
`max.length`	Maximum sequence length for embeddings
`EGA.model`	EGA network model ("glasso", "TMFG", or NULL)
`EGA.algorithm`	EGA algorithm ("walktrap", "leiden", "louvain")
`EGA.uni.method`	EGA unidimensionality method ("louvain", "expand", "LE")
`embeddings.only`	Whether to stop after embeddings
`plot`	Whether to show plots
`silently`	Whether to suppress messages

Value

A list of all validated parameters ready for local GENIE execution

Index