# Semantic network analysis for Content Validity

Frederico Pedrosa

2025-05-03

## Semantic network analysis for item validity

This syntax provides an example of how to use the transformer architecture to ensure strong content validity with an AI tool. We apply the AI-GENIE framework (Lasalandra et al., 2025) to represent the thesis of music therapist Antônio Bruno Verga Pinto (2024). The framework is used to evaluate item constructs, which were assessed by experts for children aged 3 to 6 years.

The theoretical dimensions are:

- SENSORIMOTOR
- SENSORIMOTOR BODY AWARENESS
- SENSORIMOTOR WITH OBJECTS
- SENSORIMOTOR WITH ENVIRONMENT
- RECEPTIVE COMMUNICATION
- EXPRESSIVE COMMUNICATION
- ORAL MOTOR SKILLS
- COGNITIVE ATTENTION/PERCEPTION
- COGNITIVE MEMORY/EXECUTIVE FUNCTIONS
- COGNITIVE ACADEMIC SKILLS
- VISUAL
- SOCIOEMOTIONAL PARTICIPATION
- SOCIOEMOTIONAL TURN-TAKING
- SOCIOEMOTIONAL RELATIONAL SKILLS

### 1. Load libraries and Preprocess Items

```r
raw_items <- phrases

# Define indices of items to be removed based on prior manual review
# Categories: non-musical, problematic (validity/clarity issues), duplicates/redundant
initial_item_removal_indices <- c(
  # Non-musical content items
  6, 10, 11, 17, 21, 23, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 47,
  48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 87, 88, 89, 91, 92, 133, 139, 145, 172:174, 179:186,
  # Problematic items (ambiguity, difficult interpretation, etc.)
  7, 15, 65, 68, 72, 77, 82, 93, 94, 95:116, 124, 131, 137, 138, 149:170,
  # Duplicate or highly redundant items identified manually
  20, 95, 125:128, 193, 194, 196, 199:206
)
```

```r
# Remove the specified items
filtered_items <- raw_items[-initial_item_removal_indices]

# Identify and remove any remaining exact duplicate phrases
unique_phrases <- unique(filtered_items)
n_unique_phrases <- length(unique_phrases)

print(paste("Number of unique phrases after initial filtering and deduplication:", n_unique_phrases))
```

```
## [1] "Number of unique phrases after initial filtering and deduplication: 80"
```

### 2. Generate Sentence Embeddings

```r
# Specify the dedicated Python virtual environment for sentence-transformers
# Adjust the path to your virtual environment if necessary
use_virtualenv("~/Brysa/.venv_st_ega", required = TRUE)

# Import the sentence-transformers library
# delay_load = FALSE ensures it loads immediately to catch errors early
st_module <- import("sentence_transformers", delay_load = FALSE)
SentenceTransformer <- st_module$SentenceTransformer

# Define the pre-trained sentence embedding model
# 'paraphrase-multilingual-mpnet-base-v2' is suitable for multiple languages
embedding_model_name <- 'paraphrase-multilingual-mpnet-base-v2'

# Load the sentence transformer model
st_model <- SentenceTransformer(embedding_model_name)

# Encode the unique phrases into numerical embeddings
phrase_embeddings_matrix <- st_model$encode(unique_phrases)
print("Embeddings generated.")
```

```
## [1] "Embeddings generated."
```

### 3. Format Data for EGAnet

```r
# Following the AI-GENIE paper: treat items as variables (columns) and
# embedding dimensions as observations (rows).

# Transpose the matrix: Dimensions x Items
embeddings_matrix_transposed <- t(phrase_embeddings_matrix)

# Convert to a data frame suitable for EGAnet
embeddings_df_transposed <- as.data.frame(embeddings_matrix_transposed)

# Assign meaningful column names (representing the items)
item_colnames <- paste0("i", 1:n_unique_phrases)
```

```
colnames(embeddings_df_transposed) <- item_colnames

# Verify dimensions and structure
print("Dimensions of the data frame for EGA (Embedding Dimensions x Items):")
```

```
## [1] "Dimensions of the data frame for EGA (Embedding Dimensions x Items):"
```

```
print(dim(embeddings_df_transposed))
```

```
## [1] 768  80
```

### 4. Iterative Item Reduction using UVA and bootEGA

```
# Set seed for reproducibility of bootstrap analysis
set.seed(123)

# --- Iteration 1 ---
print("--- Starting Iteration 1 of UVA and bootEGA ---")
```

```
## [1] "--- Starting Iteration 1 of UVA and bootEGA ---"
```

```
current_data_iter1 <- embeddings_df_transposed

# Step 3 (AI-GENIE): Unique Variable Analysis to remove redundant items
# Using default settings initially (cutoff might be adjusted based on AI-GENIE paper if needed)
print("Running UVA Iteration 1...")
```

```
## [1] "Running UVA Iteration 1..."
```

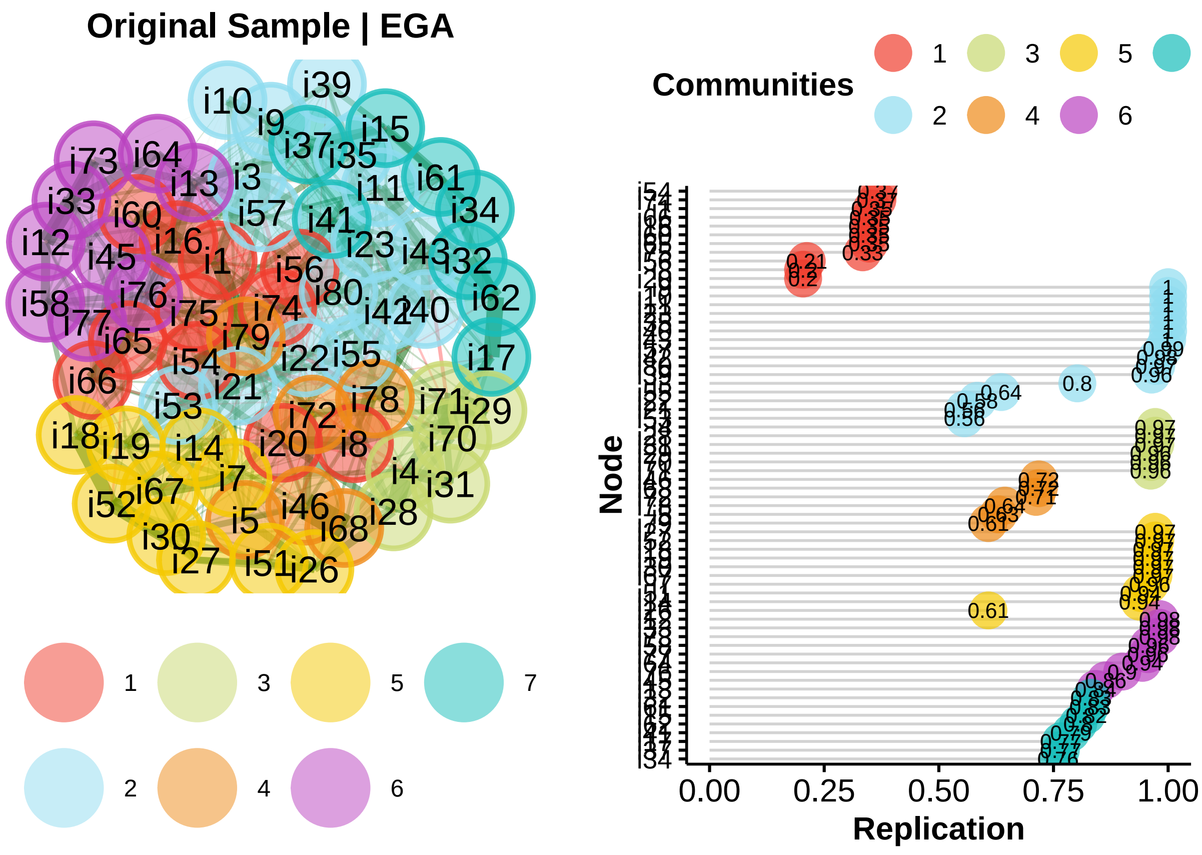
```
uva_results_iter1 <- UVA(data = current_data_iter1, verbose = FALSE)
print(paste("UVA Iteration 1 completed. Items kept:", ncol(uva_results_iter1$reduced_data)))
```

```
## [1] "UVA Iteration 1 completed. Items kept: 66"
```

```
# print(uva_results_iter1$keep_remove) # Uncomment to see details

# Store data after UVA reduction
data_after_uva_iter1 <- uva_results_iter1$reduced_data

# Step 5 (AI-GENIE): Bootstrap EGA to assess item stability
print("Running bootEGA Iteration 1...")
```

```
## [1] "Running bootEGA Iteration 1..."
```

```
bootEGA_results_iter1 <- bootEGA(
  data = data_after_uva_iter1,
  iter = 500, # Standard number of iterations, adjust if needed
  type = "parametric", # Or "resampling"
  model = "glasso", # As used later, keep consistent
  seed = 123      # Ensure reproducibility within bootEGA too
)
```



```r
print("bootEGA Iteration 1 completed.")
```

```
## [1] "bootEGA Iteration 1 completed."
```

```r
# Extract item stability proportions (empirical dimension stability)
item_stability_proportions_iter1 <- bootEGA_results_iter1$stability$item.stability$item.stability$empir

# Define the stability threshold (as per AI-GENIE paper or standard practice)
stability_threshold <- 0.75

# Identify stable and unstable items
stable_items_iter1 <- names(item_stability_proportions_iter1[item_stability_proportions_iter1 >= stabil
unstable_items_iter1 <- names(item_stability_proportions_iter1[item_stability_proportions_iter1 < stabi

print(paste("Number of stable items (>= 0.75) after Iteration 1:", length(stable_items_iter1)))
```

```
## [1] "Number of stable items (>= 0.75) after Iteration 1: 44"
```

```r
print(paste("Number of unstable items (< 0.75) after Iteration 1:", length(unstable_items_iter1)))
```

```
## [1] "Number of unstable items (< 0.75) after Iteration 1: 22"
```

```r
if (length(unstable_items_iter1) > 0) {
  print("Unstable items to be removed for next iteration:")
  print(unstable_items_iter1)
}
```

```
## [1] "Unstable items to be removed for next iteration:"
##  [1] "i1"  "i5"  "i8"  "i16" "i20" "i21" "i22" "i26" "i46" "i53" "i54" "i55"
## [13] "i56" "i60" "i65" "i66" "i68" "i72" "i74" "i75" "i78" "i79"
```

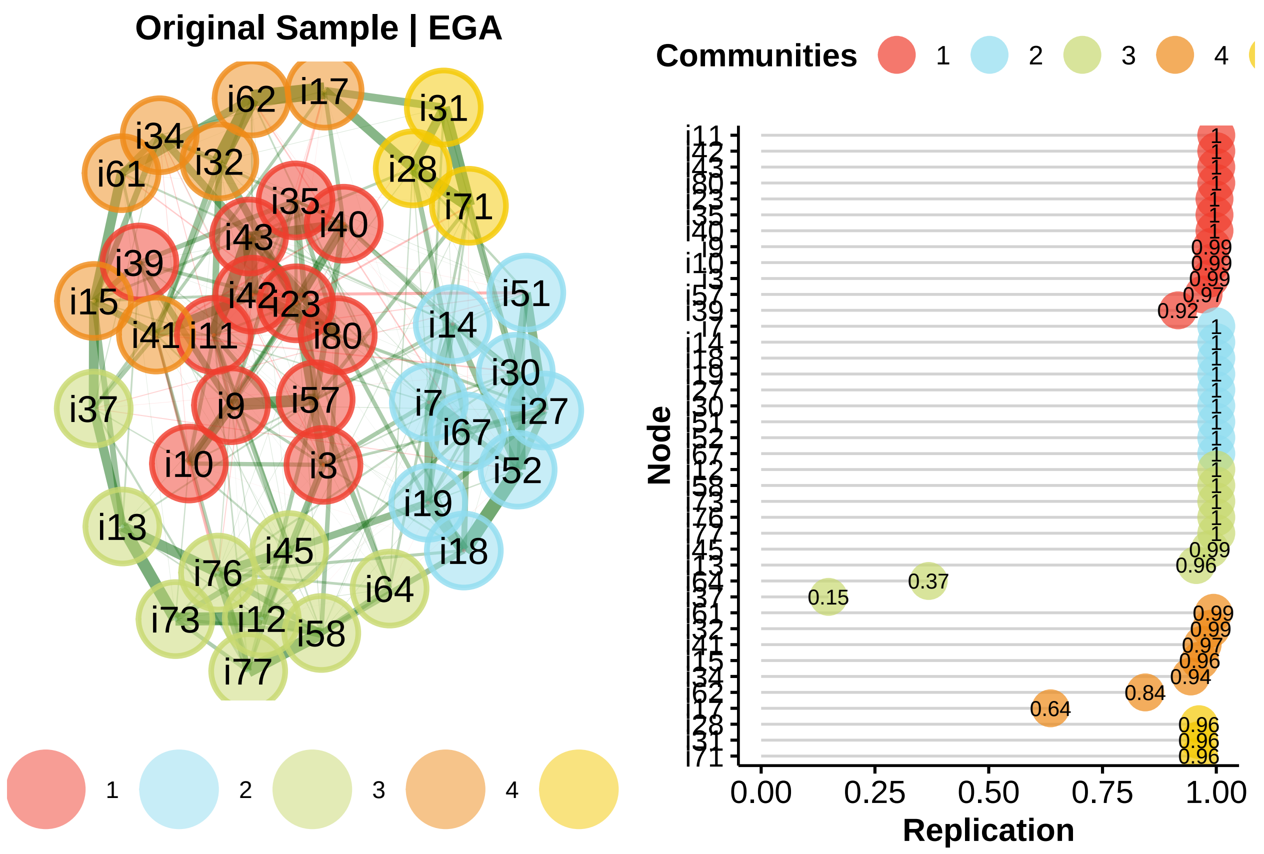
```r
# Create the dataset for the next iteration, keeping only stable items
if (length(stable_items_iter1) > 0) {
  stable_items_data_iter1 <- data_after_uva_iter1[, stable_items_iter1, drop = FALSE]
  print("Dimensions of stable data after Iteration 1 (Dimensions x Stable Items):")
  print(dim(stable_items_data_iter1))
} else {
  stop("No stable items found after Iteration 1. Stopping.")
}
```

```
## [1] "Dimensions of stable data after Iteration 1 (Dimensions x Stable Items):"
## [1] 768  44
```

```r


# Check if there were unstable items removed in the previous step
if (length(unstable_items_iter1) == 0) {
  print("--- No unstable items removed in Iteration 1. Skipping further iterations. ---")
  final_stable_data <- stable_items_data_iter1
  # We still need a final EGA run on this stable set
  print("Running final EGA on stable data from Iteration 1...")
  final_ega_run <- EGA(data = final_stable_data, model = "glasso", verbose = FALSE)
} else {
  print("--- Starting Iteration 2 of UVA and bootEGA ---")
  current_data_iter2 <- stable_items_data_iter1

  print("Running UVA Iteration 2...")
  uva_results_iter2 <- UVA(data = current_data_iter2, verbose = FALSE)
  print(paste("UVA Iteration 2 completed. Items kept:", ncol(uva_results_iter2$reduced_data)))
  data_after_uva_iter2 <- uva_results_iter2$reduced_data

  print("Running bootEGA Iteration 2...")
  bootEGA_results_iter2 <- bootEGA(
    data = data_after_uva_iter2,
    iter = 500, type = "parametric", model = "glasso", seed = 123
  )
  print("bootEGA Iteration 2 completed.")

  item_stability_proportions_iter2 <- bootEGA_results_iter2$stability$item.stability$item.stability$emp
  stable_items_iter2 <- names(item_stability_proportions_iter2[item_stability_proportions_iter2 >= stab
  unstable_items_iter2 <- names(item_stability_proportions_iter2[item_stability_proportions_iter2 < stal

  print(paste("Number of stable items (>= 0.75) after Iteration 2:", length(stable_items_iter2)))
  print(paste("Number of unstable items (< 0.75) after Iteration 2:", length(unstable_items_iter2)))
  if (length(unstable_items_iter2) > 0) {
    print("Unstable items to be removed for next iteration:")
    print(unstable_items_iter2)
  }

  if (length(stable_items_iter2) > 0) {
    stable_items_data_iter2 <- data_after_uva_iter2[, stable_items_iter2, drop = FALSE]
    print("Dimensions of stable data after Iteration 2 (Dimensions x Stable Items):")
    print(dim(stable_items_data_iter2))
  } else {
    stop("No stable items found after Iteration 2. Stopping.")
  }
}
```

**Iteration 2**

```
## [1] "--- Starting Iteration 2 of UVA and bootEGA ---"
## [1] "Running UVA Iteration 2..."
## [1] "UVA Iteration 2 completed. Items kept: 40"
## [1] "Running bootEGA Iteration 2..."
```



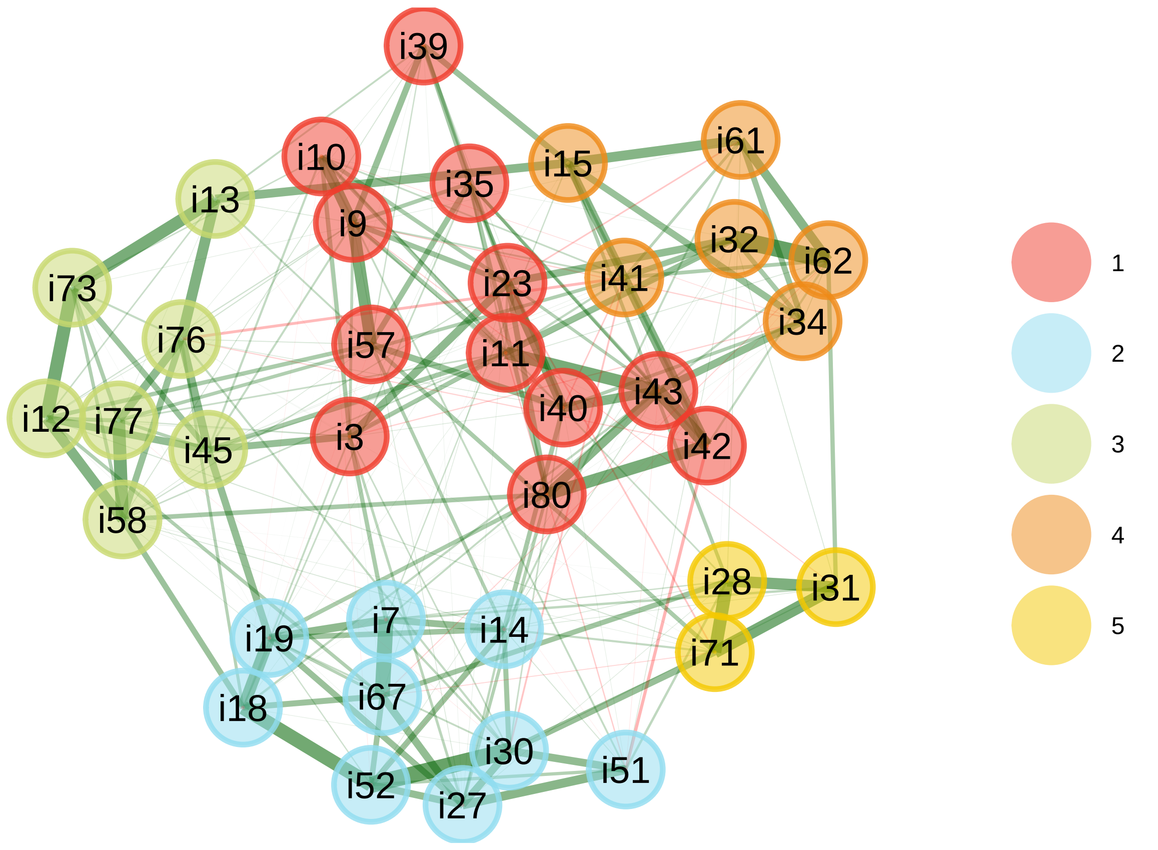
```
## [1] "bootEGA Iteration 2 completed."
## [1] "Number of stable items (>= 0.75) after Iteration 2: 37"
## [1] "Number of unstable items (< 0.75) after Iteration 2: 3"
## [1] "Unstable items to be removed for next iteration:"
## [1] "i17" "i37" "i64"
## [1] "Dimensions of stable data after Iteration 2 (Dimensions x Stable Items):"
## [1] 768  37
```

```
  if (length(unstable_items_iter2) == 0) {
  # (Code for when no items were unstable in Iteration 2 - already correct)
  print("--- No unstable items removed in Iteration 2. Preparing for final analysis. ---")
  final_stable_data <- stable_items_data_iter2
  print("Running final EGA on stable data from Iteration 2...")
  final_ega_run <- EGA(data = final_stable_data, model = "glasso", verbose = FALSE)

} else { # This else block handles the case where Iteration 3 is needed
  print("--- Starting Iteration 3 of UVA and bootEGA ---")
  current_data_iter3 <- stable_items_data_iter2

  # Note: UVA might not remove more items at this stage, but we run it for consistency.
  print("Running UVA Iteration 3...")
  uva_results_iter3 <- UVA(data = current_data_iter3, verbose = FALSE)

  # Check if the reduced_data is NULL OR has 0 columns
  if (is.null(uva_results_iter3$reduced_data) || ncol(uva_results_iter3$reduced_data) == 0) {
    # Handle the case where UVA removed all remaining items or returned NULL
    print("UVA Iteration 3 completed. No valid items remain after UVA.")
    print("Using stable data from Iteration 2 as the final dataset.")
    final_stable_data <- stable_items_data_iter2 # Use data *before* this UVA run
    print("Running final EGA on stable data from Iteration 2...")
    final_ega_run <- EGA(data = final_stable_data, model = "glasso", verbose = FALSE)
    # Skip the rest of the bootEGA/stability part for Iteration 3

  } else {
    # If UVA left items (reduced_data is not NULL and has columns), proceed
    print(paste("UVA Iteration 3 completed. Items kept:", ncol(uva_results_iter3$reduced_data)))
    data_after_uva_iter3 <- uva_results_iter3$reduced_data

    print("Running bootEGA Iteration 3...")
    # Pass the data *after* UVA (which now definitely has columns)
    bootEGA_results_iter3 <- bootEGA(
      data = data_after_uva_iter3,
      iter = 500, type = "parametric", model = "glasso", seed = 123
    )
    print("bootEGA Iteration 3 completed.")

    # Continue with stability check based on bootEGA_results_iter3
    item_stability_proportions_iter3 <- bootEGA_results_iter3$stability$item.stability$item.stability
    stable_items_iter3 <- names(item_stability_proportions_iter3[item_stability_proportions_iter3 >=
    unstable_items_iter3 <- names(item_stability_proportions_iter3[item_stability_proportions_iter3 <

    print(paste("Number of stable items (>= 0.75) after Iteration 3:", length(stable_items_iter3)))
    print(paste("Number of unstable items (< 0.75) after Iteration 3:", length(unstable_items_iter3))

    if (length(unstable_items_iter3) > 0) {
```

```r
        print("Unstable items identified in Iteration 3 (will be removed for final analysis):")
        print(unstable_items_iter3)
        warning("Items remained unstable after 3 iterations. Consider more iterations or review threshol
      }

      # Define final data based on *stable* items from THIS iteration
      if (length(stable_items_iter3) > 0) {
        # Select stable columns from the data *after* UVA filtering
        final_stable_data <- data_after_uva_iter3[, stable_items_iter3, drop = FALSE]
        print("Dimensions of final stable data after Iteration 3 (Dimensions x Stable Items):")
        print(dim(final_stable_data))
        print("Running final EGA on stable data from Iteration 3...")
        final_ega_run <- EGA(data = final_stable_data, model = "glasso", verbose = FALSE)
      } else {
        # UVA left items, but bootEGA found none stable
        stop("No stable items found by bootEGA in Iteration 3 after UVA filtering. Stopping.")
      }
    } # End of the 'if (is.null || ncol == 0)' check
  } # End of the 'else' block for Iteration 3
```

**Iteration 3**

```
## [1] "--- Starting Iteration 3 of UVA and bootEGA ---"
## [1] "Running UVA Iteration 3..."
## [1] "UVA Iteration 3 completed. No valid items remain after UVA."
## [1] "Using stable data from Iteration 2 as the final dataset."
## [1] "Running final EGA on stable data from Iteration 2..."
```



**5. Final Analysis and Reporting**

```r
# Ensure the final stable data and EGA results exist
n_final_items <- ncol(final_stable_data)
print(paste("Final number of stable items:", n_final_items))
```

```
## [1] "Final number of stable items: 37"
```

```r
print(paste("Final EGA identified", final_ega_run$n.dim, "dimensions (clusters)."))
```

```
## [1] "Final EGA identified 5 dimensions (clusters)."
```

```r
# Extract final cluster assignments for the stable items
# This is a named vector: names are item IDs (e.g., "item_5"), values are cluster numbers
final_cluster_assignments <- final_ega_run$wc

# Map the final stable item names back to their original phrases
final_stable_item_names <- names(final_cluster_assignments)

# Helper function to extract the original index from the item name "item_X"
get_index_from_item_name <- function(item_name) {
  # Remove the "item_" prefix
  num_str <- sub("^i", "", item_name)
  # Convert to integer, handle potential errors gracefully
  index <- suppressWarnings(as.integer(num_str))
  if (is.na(index)) {
    warning(paste("Could not parse index from item name:", item_name))
    return(NA)
  } else {
    return(index)
  }
}

# Apply the function to get the original indices relative to the `unique_phrases` vector
original_indices_in_unique_phrases <- sapply(final_stable_item_names,
                                             get_index_from_item_name,
                                             USE.NAMES = FALSE)

# Retrieve the original text of the final stable phrases
# Use the indices to subset the `unique_phrases` vector (which has 80 elements)
final_stable_phrases_text <- unique_phrases[original_indices_in_unique_phrases]

# --- 6. Group Final Stable Phrases by Estimated Cluster ---

# Create a temporary data frame to organize results
final_results_df <- data.frame(
  ItemName = final_stable_item_names,
  OriginalIndex = original_indices_in_unique_phrases,
  EstimatedCluster = final_cluster_assignments,
  PhraseText = final_stable_phrases_text,
  stringsAsFactors = FALSE
)

# Sort by cluster and then maybe by original index for consistent output
final_results_df <- final_results_df[order(final_results_df$EstimatedCluster, final_results_df$Original

# Use split to create a list where each element is a vector of phrases for a cluster
phrases_grouped_by_cluster <- split(final_results_df$PhraseText, final_results_df$EstimatedCluster)

# --- Display the Final Grouped Phrases ---
print("--- Final Stable Phrases Grouped by Estimated EGA Cluster ---")
```

```
## [1] "--- Final Stable Phrases Grouped by Estimated EGA Cluster ---"
```

```r
# Print in a more readable format
for (cluster_num in names(phrases_grouped_by_cluster)) {
  cat("\n=== Cluster", cluster_num, "===\n")
  # Print each phrase prefixed with "* " on a new line
  cat(paste("* ", phrases_grouped_by_cluster[[cluster_num]]), sep = "\n")
}
```

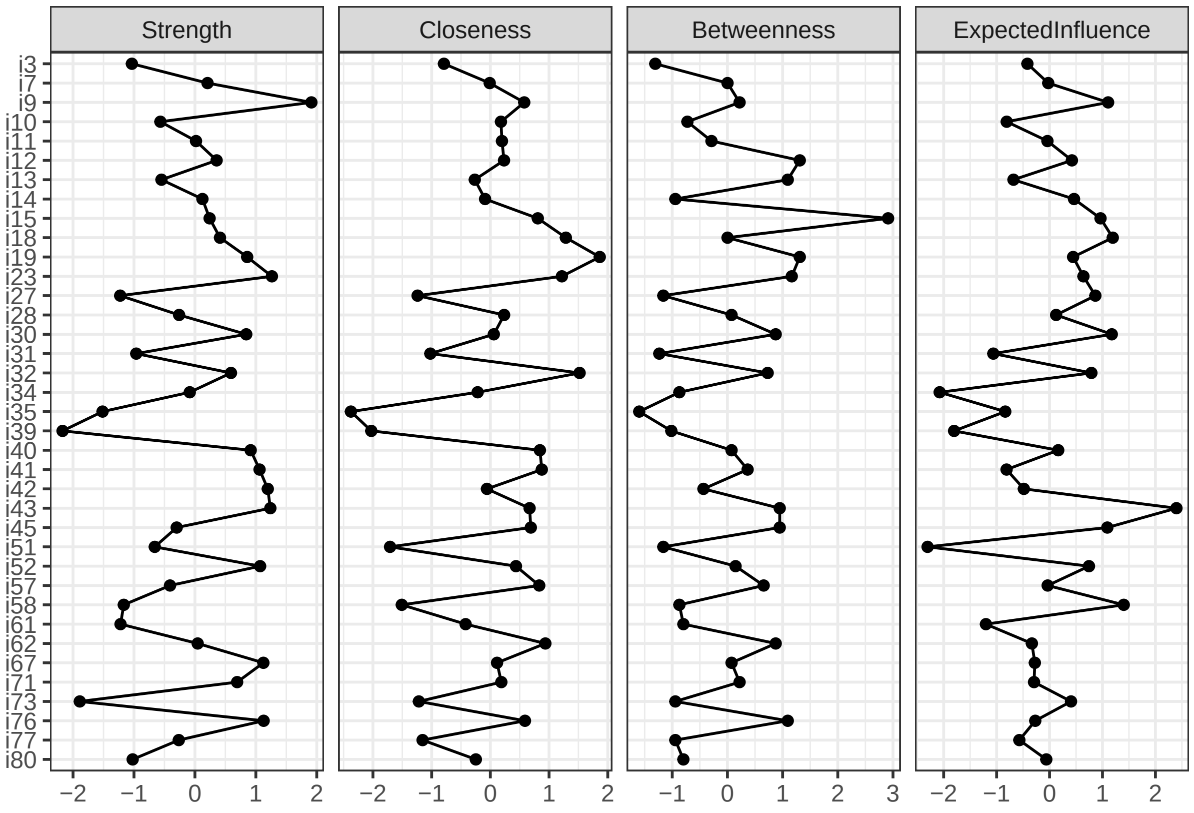
```
##
## === Cluster 1 ===
## *  Imita animais corporalmente e vocalmente durante atividades musicais?
## *  Reconhece a voz cantada masculina da voz feminina?
## *  Reconhece as 'vozes' dos personagens de desenho (ex: Mickey no tablet sem ver)?
## *  Apresenta habilidade de figura/fundo, identificando a mensagem primária na presença de sons compe
## *  Reconhece diferentes sons produzidos pela boca?
## *  O volume da voz da criança é apropriado?
## *  Articula bem o 'L' e o 'R' em encontros consonantais? (Planeta, Branco)
## *  Sustenta um som vocálico durante 7 segundos?
## *  Apresenta uma boa percepção em relação ao volume?
## *  Apresenta uma boa percepção em relação à duração do som?
## *  Fala ao microfone sem dificuldade?
## *  Percebe mudanças de volume na música?
##
## === Cluster 2 ===
## *  Locomove-se pelo ambiente transportando instrumentos musicais?
## *  Sopra um instrumento de sopro sem dificuldade?
## *  Atenção dividida: toca um instrumento e canta simultaneamente?
## *  Atenção alternada: toca os instrumentos conforme é citado na música? (ex: Loja do Mestre André)
## *  Toca o xilofone com ambas as mãos?
## *  Marcha e toca instrumentos simultaneamente sem perder o pulso?
## *  Consegue tocar as teclas do teclado individualmente com todos os dedos da mão esquerda?
## *  Consegue tocar dois instrumentos ao mesmo tempo?
## *  Compartilha instrumentos musicais?
##
## === Cluster 3 ===
## *  Canta músicas simples?
## *  Completa as frases das canções?
## *  Toca melodias simples ensinadas em encontros passados (sistema de representação perceptiva)?
## *  Canta músicas mais complexas (andamento rápido, que exige mais vocabulário)?
## *  Entende e consegue responder a perguntas simples sobre letras de músicas?
## *  Consegue criar e cantar frases musicais completas e consistentes, usando variações de ritmo e mel
## *  Usa uma variedade de palavras para criar suas próprias letras de músicas ou para descrever sentim
##
## === Cluster 4 ===
## *  Articula bem as parlendas?
## *  Demonstra habilidade para determinar se dois estímulos são iguais ou diferentes?
## *  Diferencia tempos curtos e longos (semínima e colcheias)?
## *  Apresenta uma boa percepção em relação a diferentes timbres?
## *  Faz regência simples?
## *  Apresenta controle em sua atenção seletiva (concentração)?
##
## === Cluster 5 ===
## *  Domina jogos com bambolês e dança da cadeira?
## *  Participa de atividades que envolvem escalar e descer de objetos de maneira segura e controlada?
## *  Faz coreografia sem dificuldade?
```

```
# Optional: Save the final results data frame
# write.csv(final_results_df, "final_stable_items_and_clusters.csv", row.names = FALSE)
```

## Centrality analysis

Network centrality indices (strength, closeness, betweenness, and expected influence) were estimated using the EBICglasso method. Standardized centrality scores were visualized to identify the most influential nodes.

```r
library("qgraph")
library("bootnet")


Network <- estimateNetwork(final_stable_data,
                           default = "EBICglasso", weighted = TRUE, tuning = 0.5)

centralityPlot(Network, scale = c("z-scores"),
               include = c("Strength","Closeness","Betweenness","ExpectedInfluence"),
               theme_bw = TRUE, print = TRUE,
               verbose = TRUE, weighted = TRUE,
               decreasing = T)
```



```r
print(paste("Strength centrality:", unique_phrases[9]))
```

```
## [1] "Strength centrality: Reconhece a voz cantada masculina da voz feminina?"
```

```r
print(paste("Closeness centrality:", unique_phrases[19]))
```

```
## [1] "Closeness centrality: Atenção alternada: toca os instrumentos conforme é citado na música? (ex:
```

```r
print(paste("Betweenness centrality:", unique_phrases[15]))
```

```
## [1] "Betweenness centrality: Articula bem as parlendas?"
```

```r
print(paste("Expected Influence centrality:", unique_phrases[43]))
```

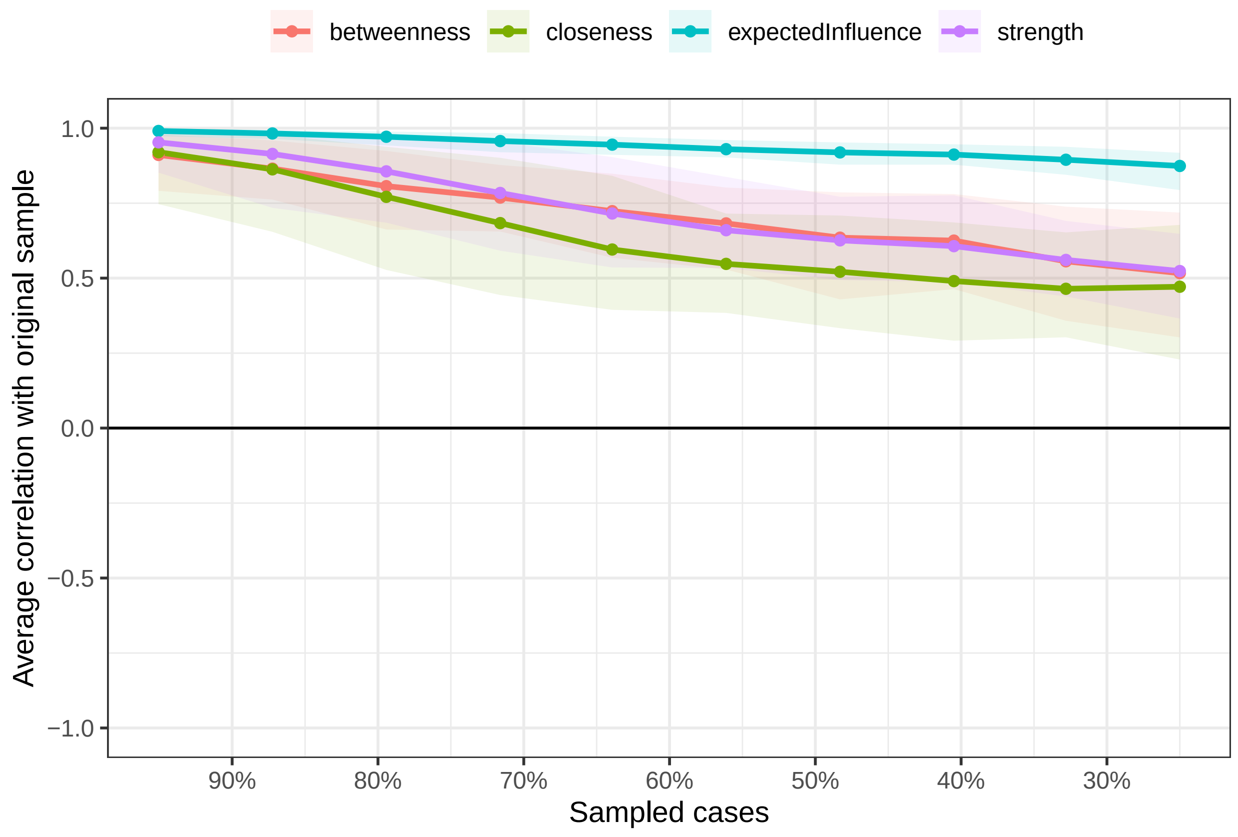
```
## [1] "Expected Influence centrality: Apresenta uma boa percepção em relação à duração do som?"
```

```r
#Estabilidade da centralidade
boot <- bootnet(Network, nBoots = 1000, type = "case",
                statistics = c("strength", "closeness", "betweenness", "expectedInfluence"))
```

**Stability of centrality measures**

```
##   |                                                                                 |

##   |                                                                                 |
```

```r
plot(boot, statistics= c("strength","closeness","betweenness","expectedInfluence"))
```



```r
#CS-coefficient
corStability(boot)
```

```
## === Correlation Stability Analysis ===
##
## Sampling levels tested:
##    nPerson Drop%   n
## 1      192  75.0  82
## 2      252  67.2 108
## 3      311  59.5  88
## 4      371  51.7 113
## 5      431  43.9  94
## 6      491  36.1  99
## 7      550  28.4 100
## 8      610  20.6 112
## 9      670  12.8 115
## 10     730   4.9  89
##
## Maximum drop proportions to retain correlation of 0.7 in at least 95% of the samples:
##
## betweenness: 0.128
##   - For more accuracy, run bootnet(..., caseMin = 0.049, caseMax = 0.206)
##
## closeness: 0.128
##   - For more accuracy, run bootnet(..., caseMin = 0.049, caseMax = 0.206)
##
## expectedInfluence: 0.75 (CS-coefficient is highest level tested)
##   - For more accuracy, run bootnet(..., caseMin = 0.672, caseMax = 1)
##
## strength: 0.206
##   - For more accuracy, run bootnet(..., caseMin = 0.128, caseMax = 0.284)
##
## Accuracy can also be increased by increasing both 'nBoots' and 'caseN'.
```

```
## R version 4.4.2 (2024-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26100)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=Portuguese_Brazil.utf8  LC_CTYPE=Portuguese_Brazil.utf8
## [3] LC_MONETARY=Portuguese_Brazil.utf8 LC_NUMERIC=C
## [5] LC_TIME=Portuguese_Brazil.utf8
##
## time zone: America/Sao_Paulo
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] bootnet_1.6       ggplot2_3.5.2     qgraph_1.9.8      psych_2.5.3
## [5] psychTools_2.4.3  EGAnet_2.3.0      reticulate_1.42.0
##
## loaded via a namespace (and not attached):
```

```
##   [1] RColorBrewer_1.1-3    rstudioapi_0.17.1     jsonlite_2.0.0
##   [4] shape_1.4.6.1         magrittr_2.0.3        jomo_2.7-6
##   [7] farver_2.1.2          nloptr_2.2.1          rmarkdown_2.29
##  [10] vctrs_0.6.5           minqa_1.2.8           base64enc_0.1-3
##  [13] rstatix_0.7.2         htmltools_0.5.8.1     progress_1.2.3
##  [16] polynom_1.4-1         plotrix_3.8-4         weights_1.0.4
##  [19] broom_1.0.8           Formula_1.2-5         mitml_0.4-5
##  [22] parallelly_1.43.0     htmlwidgets_1.6.4     plyr_1.8.9
##  [25] igraph_2.1.4          lifecycle_1.0.4       iterators_1.0.14
##  [28] pkgconfig_2.0.3       Matrix_1.7-1          R6_2.6.1
##  [31] fastmap_1.2.0         rbibutils_2.3         future_1.34.0
##  [34] digest_0.6.37         fdrtool_1.2.18        colorspace_2.1-1
##  [37] GGally_2.2.1          ellipse_0.5.0         Hmisc_5.2-3
##  [40] ggpubr_0.6.0          labeling_0.4.3        progressr_0.15.1
##  [43] nnls_1.6              gdata_3.0.1           IsingSampler_0.2.3
##  [46] abind_1.4-8           compiler_4.4.2        proxy_0.4-27
##  [49] doParallel_1.0.17     withr_3.0.2           glasso_1.11
##  [52] htmlTable_2.4.3       backports_1.5.0       carData_3.0-5
##  [55] mgm_1.2-14            ggstats_0.8.0         R.utils_2.12.3
##  [58] ggsignif_0.6.4        pan_1.9               MASS_7.3-61
##  [61] corpcor_1.6.10        rtf_0.4-14.1          gtools_3.9.5
##  [64] tools_4.4.2           pbivnorm_0.6.0        foreign_0.8-87
##  [67] future.apply_1.11.3   nnet_7.3-19           R.oo_1.27.0
##  [70] glue_1.8.0            quadprog_1.5-8        NetworkToolbox_1.4.2
##  [73] nlme_3.1-166          grid_4.4.2            checkmate_2.3.2
##  [76] cluster_2.1.6         reshape2_1.4.4        generics_0.1.3
##  [79] snow_0.4-4            gtable_0.3.6          class_7.3-22
##  [82] R.methodsS3_1.8.2     tidyr_1.3.1           sna_2.8
##  [85] data.table_1.17.0     hms_1.1.3             car_3.1-3
##  [88] foreach_1.5.2         pillar_1.10.2         stringr_1.5.1
##  [91] splines_4.4.2         dplyr_1.1.4           smacof_2.1-7
##  [94] networktools_1.5.2    lattice_0.22-6        survival_3.7-0
##  [97] tidyselect_1.2.1      pbapply_1.7-2         knitr_1.50
## [100] reformulas_0.4.0      gridExtra_2.3         IsingFit_0.4
## [103] stats4_4.4.2          xfun_0.52             stringi_1.8.7
## [106] statnet.common_4.11.0 yaml_2.3.10           boot_1.3-31
## [109] evaluate_1.0.3        codetools_0.2-20      wordcloud_2.6
## [112] tibble_3.2.1          cli_3.6.4             rpart_4.1.23
## [115] Rdpack_2.6.4          munsell_0.5.1         lavaan_0.6-19
## [118] network_1.19.0        Rcpp_1.0.14           globals_0.16.3
## [121] coda_0.19-4.1         png_0.1-8             parallel_4.4.2
## [124] prettyunits_1.2.0     jpeg_0.1-10           lme4_1.1-37
## [127] listenv_0.9.1         glmnet_4.1-8          mvtnorm_1.3-3
## [130] e1071_1.7-16          scales_1.3.0          eigenmodel_1.11
## [133] purrr_1.0.2           crayon_1.5.3          rlang_1.1.5
## [136] cowplot_1.1.3         mnormt_2.1.1          mice_3.17.0
```