Package 'RankPCA'

June 7, 2024

Type Package

Title Rank of Variables Based on Principal Component Analysis for Mixed Data Types

Version 0.1.0

Author Dr. Sandip Garai [aut, cre, cph]

Maintainer Dr. Sandip Garai <sandipnicksandy@gmail.com>

Description Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while preserving as much variability as possible. By transforming the original variables into a new set of uncorrelated variables called principal components, PCA helps in identifying patterns and simplifying the complexity of high-dimensional data. The 'RankPCA' package provides a streamlined workflow for performing PCA on datasets containing both categorical and continuous variables. It facilitates data preprocessing, encoding of categorical variables, and computes PCA to determine the optimal number of principal components based on a specified variance threshold. The package also computes composite indices for ranking observations, which can be useful for various analytical purposes. Garai, S., & Paul, R. K. (2023) <doi:10.1016/j.iswa.2023.200202>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Imports stats, caret

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-07 14:20:06 UTC

Contents

rankPCA *Rank Principal Component Analysis for Mixed Data Types*

Description

This function performs Principal Component Analysis (PCA) on datasets containing both categorical and continuous variables. It facilitates data preprocessing, encoding of categorical variables, and computes PCA to determine the optimal number of principal components based on a specified variance threshold. The function also computes composite indices for ranking observations.

Usage

```
rankPCA(data, range_cat_var, range_continuous_var, threshold)
```

Arguments

`data`	data to be analyzed.
`range_cat_var`	Range of categorical variables.
`range_continuous_var`	Range of continuous variables.
`threshold`	Threshold for cumulative variance explained.

Value

A list containing PCA results and composite index.

References

Garai, S., & Paul, R. K. (2023). Development of MCS based-ensemble models using CEEMDAN decomposition and machine intelligence. Intelligent Systems with Applications, 18, 200202, https://doi.org/10.1016/j.iswa.2023.200202.

Examples

```
# Create a sample dataset
set.seed(123)
sample_data <- data.frame(
  Category1 = sample(c("A", "B", "C"), 100, replace = TRUE),
  Category2 = sample(c("X", "Y", "Z"), 100, replace = TRUE),
  Category3 = sample(c("M", "N", "O"), 100, replace = TRUE),
  Continuous1 = rnorm(100),
  Continuous2 = runif(100, min = 0, max = 100),
  Continuous3 = rnorm(100, mean = 50, sd = 10),
  Continuous4 = rpois(100, lambda = 5),
  Continuous5 = rbinom(100, size = 10, prob = 0.5)
)
result <- rankPCA(data = sample_data,
                  range_cat_var = 1:3,
```

```
                  range_continuous_var = 4:8,
                  threshold = 80)

# Access the results
eigenvalues_pca <- result$eigenvalues_pca
pca_max_dim <- result$pca_max_dim
coordinates <- result$coordinates
eigenvalues <- result$eigenvalues
weighted_coordinates <- result$weighted_coordinates
weighted_sums <- result$weighted_sums
composite_index <- result$composite_index
loading_vectors <- result$loading_vectors
```

`variable_ranking` *Calculate Variable Ranking*

Description

This function calculates the ranking of variables based on the sum of absolute values for each row of loading vectors.

Usage

```
variable_ranking(loading_vectors)
```

Arguments

`loading_vectors`
: A matrix containing loading vectors.

Value

A data frame containing the ranked variables.

Examples

```
# Define row and column names
row_names <- c("Category1A", "Category1B", "Category1C", "Category2X", "Category2Y",
               "Category2Z", "Category3M", "Category3N", "Category3O", "Continuous1",
               "Continuous2", "Continuous3", "Continuous4", "Continuous5")

col_names <- paste0("PC", 1:8)

# Define the data matrix
loading_vectors <- matrix(c(
  0.199, 0.268, 0.189, 0.641, 0.092, 0.171, 0.079, -0.070,
  0.244, -0.371, 0.042, -0.426, 0.358, -0.070, 0.016, 0.371,
  -0.435, 0.099, -0.227, -0.216, -0.441, -0.100, -0.094, -0.294,
  0.087, -0.338, 0.458, 0.083, -0.515, -0.150, 0.007, 0.029,
  -0.473, 0.170, -0.164, 0.172, 0.296, 0.006, -0.044, 0.462,
```

```
  0.407, 0.155, -0.279, -0.261, 0.198, 0.141, 0.039, -0.510,
  0.101, -0.487, -0.465, 0.302, -0.117, 0.062, 0.036, 0.035,
  0.145, 0.546, 0.057, -0.211, -0.123, -0.325, 0.287, 0.191,
  -0.274, -0.003, 0.491, -0.134, 0.271, 0.272, -0.349, -0.245,
  0.290, 0.207, 0.001, -0.048, -0.250, -0.090, -0.275, 0.330,
  -0.134, 0.099, -0.277, -0.072, -0.180, 0.485, 0.134, 0.147,
  0.006, 0.051, -0.216, 0.007, 0.008, -0.278, -0.712, 0.004,
  0.320, 0.145, -0.061, 0.146, -0.078, 0.215, -0.414, 0.096,
  0.061, 0.044, 0.096, -0.271, -0.273, 0.603, -0.064, 0.245
), ncol = 8, byrow = TRUE)

# Assign row and column names
rownames(loading_vectors) <- row_names
colnames(loading_vectors) <- col_names

# Now you can use the loading_vectors variable in your code
print(loading_vectors)
# rank the variables
ranked_variables <- variable_ranking(loading_vectors)
print(ranked_variables)
```

Index