Package 'EpiQuestionR'

September 10, 2026

Type Package

Title Questionnaire Analysis for Epidemiology and One Health Research

Version 0.1.1

Description Provides tools for the analysis of questionnaire and survey data in epidemiological and One Health research. The package supports data preparation, reliability assessment, exploratory factor analysis, Kaiser-Meyer-Olkin assessment, parallel analysis, visualization, reporting, and export of results using a consistent analysis workflow. The methods are based on established approaches to psychometric and multivariate analysis; see Kaiser (1974) <doi:10.1007/BF02291575>, Horn (1965) <doi:10.1007/BF02289447>, and Tabachnick and Fidell (2019, ISBN:9780134790541).

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 4.3.0)

Imports ggplot2, Matrix, graphics, lavaan, psych, stats, generics, withr, tibble

Suggests broom, broom.helpers, car, corrplot, covr, dplyr, DT, flextable, forcats, gt, gtsummary, haven, janitor, knitr, leaflet, ltm, MASS, mirt, MBESS, nnet, officer, plotly, purrr, readr, readxl, rlang, rmarkdown, semPlot, sf, shiny, shinydashboard, spdep, spelling, stringr, testthat (>= 3.0.0), tidyr, tmap, e1071, jsonlite, openxlsx, reshape2, scales, yaml

VignetteBuilder knitr

URL https://github.com/vinodhpmd/EpiQuestionR

BugReports https://github.com/vinodhpmd/EpiQuestionR/issues

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Vinodh Kumar Obli Rajendran [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7232-4122>),
Keerthi Aaradhana [aut]

Maintainer Vinodh Kumar Obli Rajendran <vinodhkumar.rajendran@gmail.com>

Repository CRAN

Date/Publication 2026-09-10 09:00:18 UTC

Contents

`as.data.frame.epi_kaiser`

Convert Kaiser Result to Data Frame

Description

Converts an object of class `epi_kaiser` to a data frame containing the Kaiser criterion results.

Usage

```
## S3 method for class 'epi_kaiser'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'epi_kaiser'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An object of class epi_kaiser.
row.names	Optional row names. Currently unused.
optional	Logical. Currently unused.
...	Additional arguments, currently unused.

Value

A data frame containing the Kaiser criterion results.

---

as.data.frame.epi_map *Convert a Velicer MAP Test to a Data Frame*

Description

Converts an "epi_map" object to its component-level MAP results table.

Usage

```
## S3 method for class 'epi_map'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An object of class "epi_map".
row.names	Optional row names.
optional	Logical. Included for compatibility with base::as.data.frame().
...	Additional arguments currently ignored.

Value

A data frame containing the MAP criterion results.

as.matrix.epi_kaiser *Extract Eigenvalue Table*

Description

Extract Eigenvalue Table

Usage

```
## S3 method for class 'epi_kaiser'
as.matrix(x, ...)
```

Arguments

x	An object of class epi_kaiser.
...	Further arguments.

Value

A matrix containing the eigenvalue table stored in the table component of the epi_kaiser object. The matrix provides a tabular representation of the eigenvalue results used to summarize and interpret factor retention.

as_datatable *Interactive HTML Table*

Description

Interactive HTML Table

Usage

```
as_datatable(object, ...)
```

Arguments

object	epi_kaiser object.
...	Additional arguments passed to other methods or currently unused.

Value

DT widget.

as_flextable *Convert Kaiser Table to flextable*

Description

Convert Kaiser Table to flextable

Usage

```
as_flextable(object, ...)
```

Arguments

object	epi_kaiser object.
...	Additional arguments passed to other methods or currently unused.

Value

A flextable object.

as_gt *Convert Kaiser Table to gt*

Description

Convert Kaiser Table to gt

Usage

```
as_gt(object, ...)
```

Arguments

object	An object of class epi_kaiser.
...	Additional arguments passed to methods.

Value

A gt_tbl object.

as_kable *Convert to knitr::kable*

Description

Convert to knitr::kable

Usage

```
as_kable(object, ...)
```

Arguments

object	epi_kaiser object.
...	Additional arguments passed to other methods or currently unused.

Value

knitr kable.

as_tibble.epi_kaiser *Convert to tibble*

Description

Convert to tibble

Usage

```
## S3 method for class 'epi_kaiser'
as_tibble(x, ..., .rows = NULL, .name_repair = "check_unique", rownames = NULL)
```

Arguments

x	An object of class epi_kaiser.
...	Additional arguments passed to tibble::as_tibble().
.rows	Optional number of rows.
.name_repair	Name repair specification.
rownames	How row names should be handled.

Value

A tibble.

augment.epi_kaiser *Augment Original Dataset*

Description

Adds the retained-factor recommendation as attributes.

Usage

```
## S3 method for class 'epi_kaiser'
augment(x, data = NULL, ...)
```

Arguments

x	An object of class epi_kaiser.
data	Original dataset.
...	Additional arguments passed to the underlying function or method.

Value

A tibble.

augment.epi_parallel *Augment an EpiQuestionR Parallel Analysis*

Description

Returns component- or factor-level parallel-analysis results with additional derived quantities useful for downstream analysis, visualization, and reporting.

Usage

```
## S3 method for class 'epi_parallel'
augment(x, data = NULL, newdata = NULL, ...)
```

Arguments

x	An object of class "epi_parallel".
data	Optional data argument included for compatibility with generics::augment(). It is currently unused because parallel analysis augmentation is component-level rather than observation-level.
newdata	Optional new-data argument included for generic compatibility. It is currently unused.
...	Additional arguments. Currently unused.

Details

Unlike augmentation methods for regression models, rows in this output represent components or factors rather than individual observations.

Value

A data frame containing augmented component- or factor-level results.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)
generics::augment(result)
```

---

autoplot.epi_kaiser *Autoplot a Kaiser Criterion Analysis*

Description

Draw a scree plot for an object of class `epi_kaiser`.

Usage

```
## S3 method for class 'epi_kaiser'
autoplot(object, show_cutoff = TRUE, show_labels = TRUE, ...)
```

Arguments

`object`	An object of class `epi_kaiser`.
`show_cutoff`	Logical. Show the Kaiser eigenvalue cutoff at 1?
`show_labels`	Logical. Show eigenvalue labels?
`...`	Additional arguments, currently unused.

Value

A ggplot object.

`autoplot.epi_map` *Autoplot a Velicer MAP Test*

Description

Creates a ggplot2 visualization of a Velicer MAP test.

Usage

```
## S3 method for class 'epi_map'
autoplot(
  object,
  type = c("map", "eigenvalues"),
  highlight = TRUE,
  show_line = TRUE,
  show_label = TRUE,
  title = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

`object`	An object of class `"epi_map"`.
`type`	Character string specifying the plot type. One of `"map"` or `"eigenvalues"`.
`highlight`	Logical. If `TRUE`, highlights the recommended result.
`show_line`	Logical. If `TRUE`, displays a vertical reference line at the recommended number of components.
`show_label`	Logical. If `TRUE`, annotates the recommended number of components.
`title`	Optional plot title.
`subtitle`	Optional plot subtitle. If `NULL`, a subtitle is generated automatically.
`...`	Additional arguments currently ignored.

Details

The default `"map"` plot displays the MAP criterion against the number of partialled components and highlights the recommended dimensionality.

The `"eigenvalues"` plot creates a scree plot with a horizontal Kaiser reference line at an eigenvalue of 1.

Value

A ggplot object.

Examples

```
set.seed(123)

dat <- matrix(
  rnorm(1000),
  ncol = 10
)

fit <- map_test(dat)

ggplot2::autoplot(fit)

ggplot2::autoplot(
  fit,
  type = "eigenvalues"
)
```

---

autoplot.epi_parallel *Automatically Plot an EpiQuestionR Parallel Analysis*

Description

Creates a publication-ready **ggplot2** visualization of Horn's Parallel Analysis results.

Usage

```
## S3 method for class 'epi_parallel'
autoplot(
  object,
  type = c("parallel", "scree", "variance", "retention"),
  reference = c("criterion", "percentile", "mean", "both"),
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  legend_position = "bottom",
  base_size = 12,
  ...
)
```

Arguments

object	An object of class "epi_parallel".
type	Character string specifying the plot type. Supported values are "parallel", "variance", and "retention".
reference	Character string specifying the simulated reference series. Supported values are "criterion", "percentile", "mean", and "both".

`title`	Optional plot title.
`subtitle`	Optional plot subtitle.
`caption`	Optional plot caption.
`legend_position`	Position of the plot legend.
`base_size`	Base font size.
`...`	Additional arguments. Currently unused.

Details

Available plot types include:

`"parallel"` Observed versus simulated eigenvalue curves.

`"variance"` Percentage variance explained by each component.

`"retention"` Observed-minus-reference eigenvalue differences.

Value

A `ggplot` object.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

ggplot2::autoplot(result)

ggplot2::autoplot(
  result,
  type = "parallel",
  reference = "both"
)

ggplot2::autoplot(
  result,
  type = "variance"
)

ggplot2::autoplot(
  result,
  type = "retention"
)
```

bic *Bayesian Information Criterion*

Description

Bayesian Information Criterion

Usage

```
bic(object)
```

Arguments

object	epi_efa object.

Value

Numeric.

coef.epi_kaiser *Extract Eigenvalues*

Description

Extract Eigenvalues

Usage

```
## S3 method for class 'epi_kaiser'
coef(object, ...)
```

Arguments

object	An object of class epi_kaiser.
...	Additional arguments passed to other methods or currently unused.

Value

A named numeric vector containing the eigenvalues from the epi_kaiser object. The elements are named PC1, PC2, and so on, corresponding to successive principal components or factors. Each value represents the variance associated with the corresponding component.

coef.epi_map *Extract the Recommended Number of Components*

Description

Extracts the number of components recommended by Velicer's MAP test.

Usage

```
## S3 method for class 'epi_map'
coef(object, ...)
```

Arguments

object	An object of class "epi_map".
...	Additional arguments currently ignored.

Value

A named integer containing the recommended number of components.

Examples

```
set.seed(123)

dat <- matrix(
  rnorm(500),
  ncol = 5
)

fit <- map_test(dat)

coef(fit)
```

coef.epi_parallel *Extract Eigenvalues from an EpiQuestionR Parallel Analysis*

Description

Extracts eigenvalue-based results from an "epi_parallel" object.

Usage

```
## S3 method for class 'epi_parallel'
coef(
  object,
  type = c("observed", "simulated_mean", "simulated_percentile", "difference",
    "retained", "all"),
  ...
)
```

Arguments

object	An object of class "epi_parallel".
type	Character string specifying the values to extract. Supported values are "observed", "simulated_mean", "simulated_percentile", "difference", "retained", and "all".
...	Additional arguments passed to or from other methods.

Details

By default, the method returns observed eigenvalues. Other available outputs include mean simulated eigenvalues, percentile reference eigenvalues, differences between observed and reference eigenvalues, and retained component or factor indices.

Value

Depending on type, returns a numeric vector, integer vector, or data frame.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

coef(result)
coef(result, type = "simulated_mean")
coef(result, type = "simulated_percentile")
coef(result, type = "difference")
coef(result, type = "retained")
coef(result, type = "all")
```

communalities *Communalities*

Description

Communalities

Usage

```
communalities(object)
```

Arguments

object	epi_efa object.

Value

Numeric vector.

communality_plot *Communality Plot*

Description

Draw a publication-quality bar chart of item communalities.

Usage

```
communality_plot(
  object,
  sort = TRUE,
  horizontal = TRUE,
  fill = "#2C7FB8",
  title = "Communalities"
)
```

Arguments

object	Object of class epi_efa.
sort	Logical. Sort communalities from highest to lowest?
horizontal	Logical. Draw horizontal bars?
fill	Fill colour.
title	Plot title.

Value

A ggplot object.

Examples

```
set.seed(123)

latent <- rnorm(200)

example_data <- data.frame(
  Q1 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q2 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q3 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q4 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q5 = 0.6 * latent + rnorm(200, sd = 0.5)
)

fit <- efa(
  example_data,
  items = paste0("Q", 1:5),
  nfactors = 1
)

communality_plot(fit)
```

---

complexity *Item Complexity*

Description

Item Complexity

Usage

```
complexity(object)
```

Arguments

object	epi_efa object.

Value

Numeric vector.

`efa` *Exploratory Factor Analysis*

Description

Performs Exploratory Factor Analysis (EFA) for questionnaire data.

Usage

```
efa(
  data,
  items,
  nfactors = NULL,
  extraction = "minres",
  rotation = "oblimin",
  scores = TRUE
)
```

Arguments

`data`	Data frame.
`items`	Character vector of questionnaire items.
`nfactors`	Number of factors. If NULL, parallel analysis is used.
`extraction`	Extraction method. One of "minres","pa","ml","uls","gls","wls".
`rotation`	Rotation method. One of "varimax","promax","oblimin","quartimin", "none".
`scores`	Calculate factor scores?

Details

Features:

- KMO Measure
- Bartlett Test
- Parallel Analysis
- Automatic Factor Selection
- Multiple Extraction Methods
- Multiple Rotations

Value

Object of class `epi_efa`

Examples

```
data(questionnaire_data)

efa(
  questionnaire_data,
  items = paste0("Q", 1:5),
  nfactors = 1
)
```

`efa_fit` *EFA Fit Statistics*

Description

EFA Fit Statistics

Usage

```
efa_fit(object)
```

Arguments

`object`	epi_efa object.

Value

Data frame.

`eigenvalues` *Eigenvalues*

Description

Eigenvalues

Usage

```
eigenvalues(x, ...)
```

Arguments

`x`	An object of class `epi_kaiser`.
`...`	Additional arguments passed to methods.

Value

A numeric vector containing the eigenvalues from the `epi_kaiser` object. Each element represents the amount of variance associated with the corresponding component or factor and is used in the Kaiser criterion for factor retention.

`eigenvalue_plot` *Eigenvalue Plot*

Description

Publication-quality eigenvalue plot.

Usage

```
eigenvalue_plot(object, criterion = 1)
```

Arguments

`object`	epi_efa object.
`criterion`	Kaiser cutoff.

Value

ggplot object.

`export_kaiser_csv` *Export Kaiser Results to CSV*

Description

Exports the Kaiser criterion results to a CSV file.

Usage

```
export_kaiser_csv(
  object,
  file = tempfile(pattern = "kaiser_results_", fileext = ".csv"),
  digits = object$digits
)
```

Arguments

`object`	An object of class `epi_kaiser`.
`file`	Character. Output CSV filename.
`digits`	Number of decimal places.

Value

Invisibly returns the normalized output filename.

---

export_kaiser_excel *Export Kaiser Results to Excel*

Description

Exports the Kaiser criterion results to an Excel workbook.

Usage

```
export_kaiser_excel(
  object,
  file = tempfile(pattern = "kaiser_results_", fileext = ".xlsx"),
  digits = 3
)
```

Arguments

object	An object of class epi_kaiser.
file	Character. Output Excel filename.
digits	Number of decimal places.

Value

Invisibly returns the normalized output filename.

---

export_kaiser_json *Export Kaiser Results to JSON*

Description

Export Kaiser Results to JSON

Usage

```
export_kaiser_json(
  object,
  file = tempfile(pattern = "kaiser_results_", fileext = ".json")
)
```

Arguments

object	epi_kaiser object.
file	Output JSON filename.

Value

Invisibly returns the filename.

`export_kaiser_markdown` *Export Markdown Report*

Description

Export Markdown Report

Usage

```
export_kaiser_markdown(
  object,
  file = tempfile(pattern = "kaiser_report_", fileext = ".md"),
  digits = object$digits
)
```

Arguments

`object`	epi_kaiser object.
`file`	Output markdown filename.
`digits`	Number of decimal places.

Value

Invisibly returns the filename.

`export_kaiser_yaml` *Export Kaiser Results to YAML*

Description

Export Kaiser Results to YAML

Usage

```
export_kaiser_yaml(
  object,
  file = tempfile(pattern = "kaiser_results_", fileext = ".yml")
)
```

Arguments

object	epi_kaiser object.
file	Output YAML filename.

Value

Invisibly returns the filename.

---

export_map *Export Velicer MAP Test Results*

Description

Exports results from a Velicer Minimum Average Partial (MAP) test to a file.

Usage

```
export_map(x, file, format = NULL, digits = 4L, overwrite = FALSE, ...)
```

Arguments

x	An object of class "epi_map".
file	Character string specifying the output file path.
format	Character string specifying the export format. One of "csv", "xlsx", or "txt". If NULL, the format is inferred from the file extension.
digits	Number of decimal places used for numeric results.
overwrite	Logical. If TRUE, an existing file may be overwritten.
...	Additional arguments passed to the corresponding export method.

Details

Supported formats include CSV, Excel, and plain-text reports.

Value

The normalized path to the exported file, invisibly.

Examples

```
fit <- map_test(USArrests)

csv_file <- tempfile(fileext = ".csv")
xlsx_file <- tempfile(fileext = ".xlsx")
txt_file <- tempfile(fileext = ".txt")

export_map(
  fit,
  file = csv_file
)

export_map(
  fit,
  file = xlsx_file
)

export_map(
  fit,
  file = txt_file
)

file.exists(csv_file)
file.exists(xlsx_file)
file.exists(txt_file)
```

---

export_map_csv *Export Velicer MAP Results to CSV*

Description

Exports the component-level MAP criterion table to a CSV file.

Usage

```
export_map_csv(x, file, digits = 4L, overwrite = FALSE, row.names = FALSE, ...)
```

Arguments

x	An object of class "epi_map".
file	Character string specifying the output CSV file.
digits	Number of decimal places used for MAP values.
overwrite	Logical. If TRUE, overwrite an existing file.
row.names	Logical. Passed to utils::write.csv().
...	Additional arguments passed to utils::write.csv().

Value

The normalized output path, invisibly.

Examples

```
fit <- map_test(USArrests)

file <- tempfile(fileext = ".csv")

export_map_csv(
  fit,
  file
)
```

`export_map_excel` *Export Velicer MAP Results to Excel*

Description

Exports MAP analysis results to an Excel workbook containing separate worksheets for the analysis summary, MAP criterion results, eigenvalues, settings, and metadata.

Usage

```
export_map_excel(x, file, digits = 4L, overwrite = FALSE, ...)
```

Arguments

`x`	An object of class `"epi_map"`.
`file`	Character string specifying the output `.xlsx` file.
`digits`	Number of decimal places used for numeric values.
`overwrite`	Logical. If `TRUE`, overwrite an existing file.
`...`	Additional arguments reserved for future extensions.

Value

The normalized output path, invisibly.

Examples

```
fit <- map_test(USArrests)

file <- tempfile(fileext = ".xlsx")

export_map_excel(
  fit,
  file
)
```

`export_map_report` *Export a Velicer MAP Narrative Report*

Description

Writes a publication-ready textual report of a Velicer MAP analysis to a plain-text file.

Usage

```
export_map_report(
  x,
  file,
  style = c("standard", "apa", "brief"),
  digits = 4L,
  overwrite = FALSE,
  ...
)
```

Arguments

`x`	An object of class `"epi_map"`.
`file`	Character string specifying the output text file.
`style`	Reporting style passed to `report_map()`.
`digits`	Number of decimal places used for the MAP value.
`overwrite`	Logical. If `TRUE`, overwrite an existing file.
`...`	Additional arguments passed to `report_map()`.

Value

The normalized output path, invisibly.

Examples

```
fit <- map_test(USArrests)

file <- tempfile(fileext = ".txt")

export_map_report(
  fit,
  file
)
```

`export_parallel` *Export an EpiQuestionR Parallel Analysis Object*

Description

Saves a complete `"epi_parallel"` object to an RDS file. This preserves the full object structure, including results, settings, metadata, matrices, diagnostics, and other stored elements.

Usage

```
export_parallel(x, file, compress = TRUE)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`file`	Character scalar specifying the destination RDS file.
`compress`	Compression method passed to `saveRDS()`.

Value

Invisibly returns the normalized output file path.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

export_parallel(
  result,
```

```
  file = tempfile(fileext = ".rds")
)
```

`export_parallel_report` *Export a Parallel Analysis Report*

Description

Generates and exports a text report from an `"epi_parallel"` object.

Usage

```
export_parallel_report(x, file, digits = 3, overwrite = FALSE)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`file`	Character scalar specifying the destination text file.
`digits`	Number of decimal places used in reported numeric values.
`overwrite`	Logical. Whether an existing file may be overwritten.

Details

The exported report contains methods, results, interpretation, and recommendation sections.

Value

Invisibly returns the normalized output file path.

Examples

```
set.seed(123)
dat <- data.frame(
    item1 = rnorm(30),
    item2 = rnorm(30),
    item3 = rnorm(30),
    item4 = rnorm(30),
    item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

tmp <- tempfile(fileext = ".txt")
export_parallel_report(
    result,
    file = tmp
)
file.exists(tmp)
```

`export_parallel_tables` *Export Parallel Analysis Tables*

Description

Exports tables generated from an `"epi_parallel"` object as CSV files.

Usage

```
export_parallel_tables(
  x,
  path = ".",
  prefix = "parallel_analysis",
  digits = 3,
  include_variance = TRUE,
  include_retention = TRUE,
  create_dir = TRUE,
  overwrite = FALSE
)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`path`	Directory where CSV files will be written.
`prefix`	Character prefix used for exported file names.
`digits`	Number of decimal places used for numeric output.
`include_variance`	Logical. Export the variance table.
`include_retention`	Logical. Export the retention table.
`create_dir`	Logical. Create `path` if it does not exist.
`overwrite`	Logical. Whether existing files may be overwritten.

Details

The function can export the analysis summary, component-level parallel analysis results, variance-explained results, and retention decisions.

Value

Invisibly returns a named character vector containing exported file paths.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

export_parallel_report(
  result,
  file = tempfile(fileext = ".txt")
)
```

---

factor_correlation_plot

Factor Correlation Heatmap

Description

Draw a heatmap of the factor correlation matrix (Phi).

Usage

```
factor_correlation_plot(
  object,
  digits = 2,
  low = "#2166AC",
  mid = "white",
  high = "#B2182B"
)
```

Arguments

`object`	Object of class `epi_efa`.
`digits`	Number of decimal places.
`low`	Colour for negative correlations.
`mid`	Colour for zero.
`high`	Colour for positive correlations.

Details

This plot is only available for oblique rotations such as `"oblimin"`, `"promax"`, `"quartimin"`, etc.

Value

ggplot object.

---

factor_loadings *Extract Factor Loadings*

Description

Extract Factor Loadings

Usage

```
factor_loadings(object)
```

Arguments

object	epi_efa object.

Value

Matrix of factor loadings.

---

factor_scores *Factor Scores*

Description

Factor Scores

Usage

```
factor_scores(object)
```

Arguments

object	epi_efa object.

Value

Data frame.

glance.epi_kaiser *Glance at a Kaiser Criterion Object*

Description

Returns a one-row summary suitable for model comparison.

Usage

```
## S3 method for class 'epi_kaiser'
glance(x, ...)
```

Arguments

x	An object of class epi_kaiser.
...	Additional arguments passed to the underlying function or method.

Value

A tibble.

glance.epi_map *Glance at a Velicer MAP Test*

Description

Returns a one-row summary of a Velicer Minimum Average Partial (MAP) test.

Usage

```
## S3 method for class 'epi_map'
glance(x, ...)
```

Arguments

x	An object of class "epi_map".
...	Additional arguments currently ignored.

Value

A one-row data frame containing the recommended number of components, minimum MAP value, criterion, number of observations, number of variables, input type, correlation method, and maximum number of components evaluated.

Examples

```
set.seed(123)
dat <- matrix(rnorm(1000), ncol = 10)
fit <- map_test(dat)
generics::glance(fit)
```

`glance.epi_parallel` *Glance at an EpiQuestionR Parallel Analysis*

Description

Returns a one-row summary of Horn's Parallel Analysis, including dataset dimensions, simulation settings, retention criterion, and the recommended number of retained components or factors.

Usage

```
## S3 method for class 'epi_parallel'
glance(x, ...)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`...`	Additional arguments. Currently unused.

Value

A one-row data frame containing model-level summary information.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)
generics::glance(result)
```

kaiser_caption *APA Caption*

Description

APA Caption

Usage

```
kaiser_caption(object)
```

Arguments

object	epi_kaiser object.

Value

Character string.

kaiser_criterion *Kaiser Criterion for Factor Retention*

Description

Determines the recommended number of factors using the Kaiser criterion (retain factors with eigenvalues greater than 1).

Usage

```
kaiser_criterion(
  data,
  items = NULL,
  correlation = "auto",
  use = "pairwise.complete.obs",
  digits = 3,
  verbose = TRUE
)
```

Arguments

data	A data frame or numeric matrix.
items	Character vector of variable names. If NULL, all variables are used.
correlation	Correlation method. One of "auto", "pearson", "spearman", "kendall", "polychoric", "tetrachoric", or "mixed".
use	Missing-value handling passed to cor.
digits	Number of digits to print.
verbose	Logical.

Value

An object of class `epi_kaiser`.

Examples

```
data(bfi, package = "psych")

dat <- bfi[
  complete.cases(bfi[, 1:6]),
  1:6
]

res <- kaiser_criterion(
  dat[1:200, ],
  verbose = FALSE
)

print(res)
summary(res)
```

---

kaiser_footnote *Kaiser Table Footnote*

Description

Kaiser Table Footnote

Usage

```
kaiser_footnote(object)
```

Arguments

`object`	epi_kaiser object.

Value

Character string.

kaiser_notes *Table Notes*

Description

Table Notes

Usage

```
kaiser_notes(object)
```

Arguments

object	epi_kaiser object.

Value

Character vector.

kaiser_summary_table *Kaiser Summary Table*

Description

Kaiser Summary Table

Usage

```
kaiser_summary_table(object)
```

Arguments

object	epi_kaiser object.

Value

data.frame

kaiser_table *Kaiser Criterion Table*

Description

Create a publication-ready table summarizing the Kaiser criterion results.

Usage

```
kaiser_table(object, digits = object$digits, retain_label = "Yes")
```

Arguments

object	An object of class epi_kaiser.
digits	Number of decimal places.
retain_label	Label for retained components.

Value

A data.frame.

loading_heatmap *Factor Loading Heatmap*

Description

Draws a publication-quality heatmap of factor loadings.

Usage

```
loading_heatmap(object, cutoff = 0.3, absolute = FALSE, digits = 2)
```

Arguments

object	Object of class epi_efa
cutoff	Minimum loading to display.
absolute	Logical. Plot absolute loadings?
digits	Number of decimals.

Value

ggplot object.

Examples

```
set.seed(123)

latent <- rnorm(200)

example_data <- data.frame(
  Q1 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q2 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q3 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q4 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q5 = 0.6 * latent + rnorm(200, sd = 0.5)
)

fit <- efa(
  example_data,
  items = paste0("Q", 1:5),
  nfactors = 1
)

loading_heatmap(fit)
```

---

load_kaiser *Load Kaiser Object*

Description

Load Kaiser Object

Usage

```
load_kaiser(file)
```

Arguments

file	RDS filename.

Value

Object of class `epi_kaiser`.

`map_analysis_example` *Example Velicer MAP Analysis*

Description

Demonstrates a complete workflow for dimensionality assessment using Velicer's Minimum Average Partial (MAP) test.

Usage

```
map_analysis_example()
```

Details

The example generates simulated questionnaire data containing an underlying latent component structure and applies both the original and revised MAP criteria.

Value

A named list containing:

data Simulated questionnaire data.

original Results from the original MAP criterion.

revised Results from the revised MAP criterion.

summary A comparison table of recommendations.

Examples

```
example <- map_analysis_example()

example$original

summary(example$original)

map_table(example$original)

map_eigenvalue_table(example$original)

plot(example$original)

report_map(example$original)
```

map_eigenvalue_table *Extract an Eigenvalue Table from a Velicer MAP Test*

Description

Returns eigenvalues and explained-variance information from an "epi_map" object.

Usage

```
map_eigenvalue_table(x, digits = 4L, percentage = FALSE)
```

Arguments

x	An object of class "epi_map".
digits	Number of decimal places used for rounding.
percentage	Logical. If TRUE, variance proportions are expressed as percentages.

Value

A data frame containing component numbers, eigenvalues, variance proportions, and cumulative variance.

Examples

```
fit <- map_test(USArrests)
map_eigenvalue_table(fit)
```

map_interpretation *Interpret a Velicer MAP Test*

Description

Generates a concise textual interpretation of the dimensionality recommendation from Velicer's MAP test.

Usage

```
map_interpretation(x, include_value = TRUE, digits = 4L)
```

Arguments

x	An object of class "epi_map".
include_value	Logical. If TRUE, includes the minimum MAP criterion value in the interpretation.
digits	Number of decimal places used for the MAP value.

Value

A character string containing the interpretation.

Examples

```
fit <- map_test(USArrests)

map_interpretation(fit)
```

`map_report_table` *Create a Publication Summary Table for Velicer's MAP Test*

Description

Creates a compact one-row summary suitable for manuscripts, supplementary tables, or export.

Usage

```
map_report_table(x, digits = 4L)
```

Arguments

`x`	An object of class `"epi_map"`.
`digits`	Number of decimal places used for numeric values.

Value

A one-row data frame summarizing the MAP analysis.

Examples

```
fit <- map_test(USArrests)

map_report_table(fit)
```

map_table *Extract a Velicer MAP Results Table*

Description

Extracts a publication-friendly table of MAP criterion values from an "epi_map" object.

Usage

```
map_table(x, digits = 4L, labels = TRUE)
```

Arguments

x	An object of class "epi_map".
digits	Number of decimal places used to round the MAP criterion.
labels	Logical. If TRUE, adds human-readable recommendation labels.

Value

A data frame containing the MAP results.

Examples

```
fit <- map_test(USArrests)
map_table(fit)
```

map_test *Velicer's Minimum Average Partial (MAP) Test*

Description

Performs Velicer's Minimum Average Partial (MAP) test to estimate the optimal number of components to retain in dimensionality assessment.

Usage

```
map_test(
  data,
  method = c("original", "revised"),
  correlation = c("pearson", "spearman", "kendall"),
  use = "pairwise.complete.obs",
  is_corr = FALSE,
  max_components = NULL,
  tolerance = sqrt(.Machine$double.eps),
  ...
)
```

Arguments

`data`	A numeric data frame or matrix containing observations in rows and variables in columns. Alternatively, a correlation matrix when `is_corr = TRUE`.
`method`	Character string specifying the MAP criterion. One of `"original"` or `"revised"`. The original criterion uses squared partial correlations, whereas the revised criterion uses fourth-power partial correlations.
`correlation`	Character string specifying the correlation coefficient used when `data` contains raw observations. Currently `"pearson"` and `"spearman"` are supported.
`use`	Character string specifying the missing-value handling method passed to `stats::cor()`. Common options include `"everything"`, `"complete.obs"`, and `"pairwise.complete.obs"`.
`is_corr`	Logical. If `TRUE`, `data` is treated as a correlation matrix. If `FALSE`, a correlation matrix is computed from the raw data.
`max_components`	Optional integer specifying the maximum number of components to evaluate. By default, up to `p - 1` components are considered, where `p` is the number of variables.
`tolerance`	Numeric tolerance used when handling very small eigenvalues and numerical instability.
`...`	Additional arguments reserved for future extensions.

Details

The MAP procedure sequentially partials principal components from a correlation matrix. After each component is removed, the average squared partial correlation among variables is calculated. The number of components associated with the minimum average partial correlation is selected as the recommended dimensionality.

A revised MAP criterion based on the average fourth power of the partial correlations is also available.

Velicer's MAP procedure begins with the original correlation matrix and computes an average measure of the off-diagonal correlations. Principal components are then sequentially removed. At each step, a residual covariance matrix is reconstructed and standardized to obtain a partial correlation matrix.

For the original MAP criterion, the statistic at step k is the mean squared off-diagonal partial correlation.

For the revised MAP criterion, the statistic is the mean fourth power of the off-diagonal partial correlations.

The recommended number of components corresponds to the step with the smallest MAP statistic. Step zero represents the unpartialled correlation matrix and therefore allows the procedure to recommend zero components.

The MAP test is commonly used alongside parallel analysis when determining the dimensionality of questionnaire and psychometric data.

Value

An object of class `"epi_map"` containing:

n_components Recommended number of components.

map_value Minimum MAP statistic.

criterion MAP criterion used.

results A data frame containing the number of partialled components and corresponding MAP statistics.

eigenvalues Eigenvalues of the input correlation matrix.

correlation_matrix Correlation matrix used in the analysis.

settings Analysis settings.

metadata Information about the input data.

call Matched function call.

References

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. Psychometrika, 41, 321-327.

Velicer, W. F., Eaton, C. A., & Fava, J. L. (2000). Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components. In R. D. Goffin & E. Helmes (Eds.), Problems and Solutions in Human Assessment.

See Also

parallel_analysis()

Examples

```
set.seed(123)

dat <- data.frame(
  item1 = rnorm(300),
  item2 = rnorm(300),
  item3 = rnorm(300),
  item4 = rnorm(300),
  item5 = rnorm(300)
)

# Original MAP
fit <- map_test(dat)

fit

# Revised MAP
fit_revised <- map_test(
  dat,
  method = "revised"
)

fit_revised
```

```
# Correlation matrix input
R <- cor(dat)

fit_cor <- map_test(
  R,
  is_corr = TRUE
)
```

`nfactors` *Number of Retained Factors*

Description

Number of Retained Factors

Usage

```
nfactors(x, ...)
```

Arguments

`x`	An object of class `epi_kaiser`.
`...`	Additional arguments passed to other methods or currently unused.

Value

A single non-negative integer giving the number of factors retained by the Kaiser criterion. For an object of class `epi_kaiser`, this value is obtained from the `nfactors` component of the object.

`parallel_analysis` *Horn's Parallel Analysis*

Description

Performs Horn's parallel analysis to estimate the number of principal components or common factors to retain. Observed eigenvalues are compared with eigenvalues obtained from simulated random datasets.

Usage

```
parallel_analysis(
  data,
  analysis = c("pca", "fa"),
 correlation = c("auto", "pearson", "spearman", "polychoric", "tetrachoric", "mixed"),
  n_iter = 1000L,
  criterion = c("percentile", "mean"),
  percentile = 95,
  extraction = "minres",
  use = "pairwise.complete.obs",
  n_cores = 1L,
  seed = NULL,
  simulate = c("normal", "permutation"),
  adjust_pd = TRUE,
  diagnostics = TRUE,
  progress = interactive(),
  verbose = TRUE,
  ...
)
```

Arguments

data	A data frame, matrix, or correlation matrix.
analysis	Character. Either "pca" or "fa".
correlation	Character. Correlation method. One of "auto", "pearson", "spearman", "polychoric", "tetrachoric", or "mixed".
n_iter	Integer. Number of random datasets to simulate.
criterion	Character. Retention criterion used to compare observed and simulated eigenvalues. Either "percentile" or "mean".
percentile	Numeric. Percentile of simulated eigenvalues used as the retention threshold. Must be strictly between 0 and 100.
extraction	Character. Factor extraction method passed to psych::fa() when analysis = "fa".
use	Character. Missing-value handling passed to stats::cor().
n_cores	Integer. Number of CPU cores. Values greater than 1 enable multicore execution on supported platforms.
seed	Integer or NULL. Random seed used for reproducible simulations.
simulate	Character. Simulation strategy. Either "normal" or "permutation".
adjust_pd	Logical. If TRUE, non-positive-definite correlation matrices are adjusted.
diagnostics	Logical. If TRUE, diagnostic information is calculated.
progress	Logical. If TRUE, display progress in serial execution.
verbose	Logical. If TRUE, print informative messages.
...	Additional arguments passed to correlation routines where appropriate.

Details

The function supports PCA and factor-analysis retention criteria, Pearson, Spearman, polychoric, tetrachoric, and mixed correlation matrices, serial and multicore simulation, reproducible random-number generation, and optional diagnostic information.

Value

An object of class `"epi_parallel"`.

`parallel_interpret` *Interpret Horn's Parallel Analysis Results*

Description

Generates a human-readable interpretation of an `"epi_parallel"` object.

Usage

```
parallel_interpret(
  x,
  style = c("brief", "publication", "detailed"),
  digits = 3
)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`style`	Character string specifying the reporting style. Supported values are `"brief"`, `"publication"`, and `"detailed"`.
`digits`	Number of decimal places used when reporting eigenvalues.

Details

Three reporting styles are available:

`"brief"` Returns a concise statement of the recommended number of dimensions.

`"publication"` Returns manuscript-ready results text.

`"detailed"` Returns a more detailed interpretation including analysis settings, retention criterion, retained indices, and recommendation.

Value

A character scalar containing the interpretation.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

parallel_interpret(result)
```

---

parallel_plot *Parallel Analysis Plot*

Description

Creates a plot comparing the observed eigenvalues from an exploratory factor analysis with simulated reference eigenvalues from Horn's parallel analysis. The plot can be used to assess the number of factors that should be retained.

Usage

```
parallel_plot(object)
```

Arguments

`object`	An object of class `"epi_efa"`.

Value

A `ggplot` object. The plot displays observed eigenvalues and simulated reference eigenvalues across components or factors. Components for which the observed eigenvalue exceeds the relevant simulated reference value are candidates for retention.

`parallel_report` *Generate a Parallel Analysis Report*

Description

Generates a structured report from an `"epi_parallel"` object suitable for manuscript preparation, thesis reporting, technical reports, or reproducible analysis pipelines.

Usage

```
parallel_report(
  x,
  digits = 3,
  include_variance = TRUE,
  include_retention = TRUE
)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`digits`	Number of decimal places used in reported numeric values.
`include_variance`	Logical. Whether to include the variance table.
`include_retention`	Logical. Whether to include the retention table.

Details

The returned object contains methods text, results text, interpretation, recommendation, analysis settings, and publication-ready tables.

Value

An object of class `"epi_parallel_report"` containing:

methods Methods-section text.

results Results-section text.

interpretation Detailed interpretation.

recommendation Recommended dimensional solution.

summary_table One-row analysis summary table.

results_table Component- or factor-level results table.

variance_table Optional variance-explained table.

retention_table Optional retention-decision table.

settings Analysis settings.

metadata Analysis metadata.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

report <- parallel_report(result)
```

---

parallel_report_text *Extract Text from a Parallel Analysis Report*

Description

Extracts selected narrative sections from an "epi_parallel_report" object.

Usage

```
parallel_report_text(
  x,
  section = c("all", "methods", "results", "interpretation", "recommendation")
)
```

Arguments

x	An object of class "epi_parallel_report".
section	Character string specifying the report section to extract. Supported values are "methods", "results", "interpretation", "recommendation", and "all".

Value

A character scalar.

parallel_retention_table

Create a Parallel Analysis Retention Table

Description

Creates a table focused on the component or factor retention decision.

Usage

```
parallel_retention_table(x, digits = 3)
```

Arguments

x	An object of class "epi_parallel".
digits	Number of decimal places.

Value

A data frame.

parallel_summary_table

Create a Parallel Analysis Summary Table

Description

Creates a compact one-row summary of the parallel analysis.

Usage

```
parallel_summary_table(x)
```

Arguments

x	An object of class "epi_parallel".

Value

A one-row data frame.

parallel_table *Create a Parallel Analysis Results Table*

Description

Extracts a clean component-level results table from an "epi_parallel" object.

Usage

```
parallel_table(x, digits = 3, include_decision = TRUE)
```

Arguments

x	An object of class "epi_parallel".
digits	Number of decimal places used for rounding numeric values.
include_decision	Logical. Include a human-readable retention decision column.

Value

A data frame containing parallel analysis results.

parallel_variance_table *Create a Variance Explained Table*

Description

Creates a table containing observed eigenvalues, percentage variance explained, cumulative variance, and retention decisions.

Usage

```
parallel_variance_table(x, digits = 2)
```

Arguments

x	An object of class "epi_parallel".
digits	Number of decimal places.

Value

A data frame.

plot.epi_efa *Plot EFA Results*

Description

Produces publication-ready EFA graphics.

Usage

```
## S3 method for class 'epi_efa'
plot(x, type = c("scree", "parallel"), ...)
```

Arguments

x	Object of class epi_efa
type	Plot type. One of: • "scree" • "parallel"
...	Additional arguments passed to the underlying function or method.

Value

A plot.

plot.epi_kaiser *Scree Plot for Kaiser Criterion*

Description

Creates a scree plot with the Kaiser cut-off (Eigenvalue = 1).

Usage

```
## S3 method for class 'epi_kaiser'
plot(
  x,
  type = c("base", "ggplot"),
  show_cutoff = TRUE,
  show_labels = TRUE,
  point_size = 19,
  line_size = 2,
  ...
)
```

Arguments

x	An object of class epi_kaiser.
type	Plot type ("base" or "ggplot").
show_cutoff	Logical; draw Eigenvalue = 1 line.
show_labels	Logical; display eigenvalue labels.
point_size	Point size.
line_size	Line width.
...	Additional arguments passed to the underlying function or method.

Value

For type = "base", no return value. The function is called for its side effect of producing a scree plot showing the eigenvalues and the Kaiser cut-off at Eigenvalue = 1. For type = "ggplot", returns a ggplot object representing the same scree plot, which can be further modified or displayed.

---

plot.epi_map *Plot a Velicer MAP Test*

Description

Creates a base R plot for an object returned by map_test().

Usage

```
## S3 method for class 'epi_map'
plot(
  x,
  type = c("map", "eigenvalues"),
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  highlight = TRUE,
  show_line = TRUE,
  show_label = TRUE,
  ...
)
```

Arguments

x	An object of class "epi_map".
type	Character string specifying the plot type. One of "map" or "eigenvalues".
main	Optional character string specifying the plot title.
xlab	Optional character string specifying the x-axis label.
ylab	Optional character string specifying the y-axis label.

highlight	Logical. If TRUE, highlights the recommended number of components.
show_line	Logical. If TRUE, draws a vertical reference line at the recommended number of components.
show_label	Logical. If TRUE, displays a text label identifying the recommended number of components.
...	Additional graphical parameters passed to graphics::plot().

Details

The default plot displays the MAP criterion across the number of partialled components. The recommended number of components is highlighted at the minimum MAP value.

An eigenvalue scree plot can also be requested.

Value

The input object x, invisibly.

Examples

```
set.seed(123)

dat <- matrix(
  rnorm(1000),
  ncol = 10
)

fit <- map_test(dat)

plot(fit)

plot(
  fit,
  type = "eigenvalues"
)
```

---

plot.epi_parallel *Plot an EpiQuestionR Parallel Analysis*

Description

Creates a base R visualization of Horn's Parallel Analysis results.

Usage

```
## S3 method for class 'epi_parallel'
plot(
  x,
  type = c("parallel", "scree", "variance", "retention"),
  reference = c("criterion", "percentile", "mean", "both"),
  main = NULL,
  xlab = "Component / Factor",
  ylab = NULL,
  legend = TRUE,
  ...
)
```

Arguments

x	An object of class "epi_parallel".
type	Character string specifying the plot type. Supported values are "parallel", "variance", and "retention".
reference	Character string specifying the simulated reference series for the parallel plot. Supported values are "criterion", "percentile", "mean", and "both".
main	Optional plot title.
xlab	Label for the x-axis.
ylab	Optional label for the y-axis.
legend	Logical. If TRUE, display a legend.
...	Additional graphical parameters passed to plotting functions.

Details

The default plot compares observed eigenvalues with simulated reference eigenvalues. Components or factors are retained when their observed eigenvalues exceed the selected simulated reference criterion.

Value

Invisibly returns x.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)
```

```
plot(result)
plot(result, type = "parallel")
plot(result, type = "variance")
plot(result, type = "retention")
plot(result, reference = "both")
```

`plot_eigenvalues` *Eigenvalue Bar Plot*

Description

Eigenvalue Bar Plot

Usage

```
plot_eigenvalues(object)
```

Arguments

`object`	epi_kaiser object.

Value

A `ggplot` object containing a bar plot of the eigenvalues from the `epi_kaiser` object. Each bar represents the eigenvalue associated with a component or factor, allowing the relative magnitude of the eigenvalues to be visually assessed.

`plot_kaiser` *Plot Dispatcher*

Description

Plot Dispatcher

Usage

```
plot_kaiser(object, which = c("scree", "eigenvalues", "variance"))
```

Arguments

`object`	epi_kaiser object.
`which`	Character string: "scree", "eigenvalues", "variance".

Value

The plot generated for the selected value of `which`. If `which = "scree"`, `which = "eigenvalues"`, or `which = "variance"`, the corresponding plot is displayed. The function returns the result invisibly, where applicable, primarily for its plotting side effect.

`plot_variance` *Variance Explained Plot*

Description

Creates a plot showing the proportion of variance explained by each retained or available component or factor.

Usage

```
plot_variance(object)
```

Arguments

`object`	An object containing the factor or component analysis results required for the plot.

Value

A plot showing the proportion of variance explained by each component or factor. The plot is generated for visual interpretation of the contribution of each component or factor and is returned invisibly, where applicable.

`print.epi_efa` *Print EFA Results*

Description

Print EFA Results

Usage

```
## S3 method for class 'epi_efa'
print(x, digits = 3, ...)
```

Arguments

`x`	Object of class `epi_efa`
`digits`	Number of decimal places.
`...`	Additional arguments passed to the underlying function or method.

Value

Prints EFA summary.

`print.epi_kaiser` *Print Kaiser Criterion Results*

Description

Print Kaiser Criterion Results

Usage

```
## S3 method for class 'epi_kaiser'
print(x, digits = x$digits, ...)
```

Arguments

`x`	An object of class `epi_kaiser`.
`digits`	Integer specifying the number of decimal places used when displaying numeric results.
`...`	Additional arguments passed to other methods or currently unused.

Value

No visible return value. The `epi_kaiser` object is returned invisibly after a formatted summary of the Kaiser criterion results is printed to the console. The printed output summarizes the eigenvalue-based factor-retention results.

`print.epi_map` *Print a Velicer MAP Test*

Description

Prints a concise summary of the results from Velicer's Minimum Average Partial (MAP) test.

Usage

```
## S3 method for class 'epi_map'
print(x, digits = 4L, ...)
```

Arguments

`x`	An object of class `"epi_map"`.
`digits`	Number of significant digits used when printing numeric results.
`...`	Additional arguments currently ignored.

Value

The input object x, invisibly.

Examples

```
set.seed(123)

dat <- data.frame(
  item1 = rnorm(300),
  item2 = rnorm(300),
  item3 = rnorm(300),
  item4 = rnorm(300),
  item5 = rnorm(300)
)

fit <- map_test(dat)
print(fit)
```

---

print.epi_parallel *Print an EpiQuestionR Parallel Analysis*

Description

Prints a concise summary of a Horn's Parallel Analysis result produced by EpiQuestionR.

Usage

```
## S3 method for class 'epi_parallel'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class "epi_parallel".
digits	Number of digits used when displaying numeric values.
...	Additional arguments passed to or from other methods.

Details

The print method reports the analysis method, number of variables, simulation settings, retention criterion, and the recommended number of components or factors.

Value

Invisibly returns x.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)
print(result)
```

---

print.epi_parallel_report

Print an EpiQuestionR Parallel Analysis Report

Description

Prints a structured report generated by `parallel_report()`.

Usage

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

Arguments

`x`	An object of class `"epi_parallel_report"`.
`...`	Additional arguments passed to or from other methods.

Value

Invisibly returns `x`.

---

print.summary.epi_map *Print a Summary of a Velicer MAP Test*

Description

Prints a detailed summary of an object returned by `summary.epi_map()`.

Usage

```
## S3 method for class 'summary.epi_map'
print(x, digits = 4L, ...)
```

Arguments

x	An object of class "summary.epi_map".
digits	Number of significant digits used when displaying numeric results.
...	Additional arguments currently ignored.

Value

The input object x, invisibly.

---

print.summary_epi_parallel *Print a Parallel Analysis Summary*

Description

Prints a formatted summary of an object returned by summary.epi_parallel().

Usage

```
## S3 method for class 'summary_epi_parallel'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class "summary_epi_parallel".
digits	Number of digits used for numeric output.
...	Additional arguments passed to or from other methods.

Value

Invisibly returns x.

---

questionnaire_data *Example Questionnaire Dataset*

Description

Simulated questionnaire data included with EpiQuestionR.

Format

A data frame with 100 rows and 5 variables:

Q1 Likert score (1-5)

Q2 Likert score (1-5)

Q3 Likert score (1-5)

Q4 Likert score (1-5)

Q5 Likert score (1-5)

Source

Simulated data

Examples

```
data(questionnaire_data)
head(questionnaire_data)
```

`report_kaiser` *Reporting Sentence*

Description

Reporting Sentence

Usage

```
report_kaiser(object)
```

Arguments

`object`	epi_kaiser object.

Value

Character string.

report_map *Report a Velicer MAP Test*

Description

Generates a publication-ready narrative report of a Velicer Minimum Average Partial (MAP) analysis.

Usage

```
report_map(
  x,
  style = c("standard", "apa", "brief"),
  digits = 4L,
  include_sample = TRUE,
  include_method = TRUE
)
```

Arguments

x	An object of class "epi_map".
style	Character string specifying the reporting style. One of "standard", "apa", or "brief".
digits	Number of decimal places used when reporting the MAP value.
include_sample	Logical. If TRUE, includes the number of observations when available.
include_method	Logical. If TRUE, includes information about the correlation and MAP methods.

Value

A character string containing a report-ready interpretation.

Examples

```
set.seed(123)

dat <- matrix(
  rnorm(1000),
  ncol = 10
)

fit <- map_test(dat)

report_map(fit)

report_map(
  fit,
  style = "apa"
```

```
)
```

`residual_matrix` *Residual Correlation Matrix*

Description

Residual Correlation Matrix

Usage

```
residual_matrix(object)
```

Arguments

`object`	epi_efa object.

Value

Matrix.

`retained_components` *Retained Components*

Description

Extract the indices of components retained by the Kaiser criterion.

Usage

```
retained_components(object)
```

Arguments

`object`	An object of class `epi_kaiser`.

Value

An integer vector containing the indices of retained components.

rmsea *RMSEA*

Description

RMSEA

Usage

```
rmsea(object)
```

Arguments

object	epi_efa object.

Value

Numeric.

rmsr *RMSR*

Description

RMSR

Usage

```
rmsr(object)
```

Arguments

object	epi_efa object.

Value

Numeric.

save_communality_plot *Save Communality Plot*

Description

Export a publication-quality communality plot.

Usage

```
save_communality_plot(
  object,
  filename = tempfile(pattern = "communality_plot_", fileext = ".tiff"),
  width = 6,
  height = 5,
  dpi = 600
)
```

Arguments

object	epi_efa object.
filename	Output filename.
width	Width in inches.
height	Height in inches.
dpi	Resolution.

Value

Invisibly returns the filename.

save_efa_plot *Save EFA Plot*

Description

Save EFA Plot

Usage

```
save_efa_plot(
  object,
  filename = tempfile(pattern = "efa_plot_", fileext = ".tiff"),
  type = "scree",
  width = 6,
  height = 5,
  dpi = 600
)
```

Arguments

object	An object of class epi_efa.
filename	Character string giving the output filename. If omitted, a temporary file is created automatically.
type	Plot type.
width	Width in inches.
height	Height in inches.
dpi	Resolution.

Value

Invisibly returns a character string containing the path to the saved plot file. The function is primarily called for its side effect of writing the selected EFA plot to the specified output file.

---

save_eigenvalue_plot *Save Eigenvalue Plot*

Description

Saves the eigenvalue plot to an image file.

Usage

```
save_eigenvalue_plot(
  object,
  filename = tempfile(pattern = "eigenvalues_", fileext = ".tiff"),
  width = 6,
  height = 5,
  dpi = 600
)
```

Arguments

object	An object of class epi_efa.
filename	Character. Output filename.
width	Numeric. Width of the saved plot in inches.
height	Numeric. Height of the saved plot in inches.
dpi	Numeric. Resolution of the saved plot in dots per inch.

Value

Invisibly returns the saved plot.

save_factor_correlation_plot *Save Factor Correlation Plot*

Description

Saves the factor correlation plot to an image file.

Usage

```
save_factor_correlation_plot(
  object,
  filename = tempfile(pattern = "factor_correlation_", fileext = ".tiff"),
  width = 6,
  height = 6,
  dpi = 600
)
```

Arguments

object	An object of class epi_efa.
filename	Character. Output filename.
width	Numeric. Width of the saved plot in inches.
height	Numeric. Height of the saved plot in inches.
dpi	Numeric. Resolution of the saved plot in dots per inch.

Value

Invisibly returns the saved plot.

save_kaiser *Save Kaiser Criterion Object*

Description

Saves the complete epi_kaiser object to an RDS file.

Usage

```
save_kaiser(
  object,
  file = tempfile(pattern = "kaiser_results_", fileext = ".rds")
)
```

Arguments

object	An object of class epi_kaiser.
file	Character. Output RDS filename.

Value

Invisibly returns the normalized output filename.

---

save_loading_heatmap *Save Loading Heatmap*

Description

Save Loading Heatmap

Usage

```
save_loading_heatmap(
  object,
  filename = tempfile(pattern = "loading_heatmap_", fileext = ".tiff"),
  width = 7,
  height = 6,
  dpi = 600
)
```

Arguments

object	epi_efa object.
filename	Character string giving the output filename. If omitted, a temporary file is created automatically.
width	Width in inches.
height	Height in inches.
dpi	Resolution.

Value

Invisibly returns the normalized path to the saved heatmap file as a character string. The function is primarily called for its side effect of writing the loading heatmap to the specified output file.

save_uniqueness_plot *Save Uniqueness Plot*

Description

Export a publication-quality uniqueness plot.

Usage

```
save_uniqueness_plot(
  object,
  filename = tempfile(pattern = "uniqueness_plot_", fileext = ".tiff"),
  width = 7,
  height = 6,
  dpi = 600
)
```

Arguments

object	Object of class epi_efa.
filename	Output filename.
width	Plot width (inches).
height	Plot height (inches).
dpi	Resolution.

Value

Invisibly returns the filename.

save_variance_explained_plot *Save Variance Explained Plot*

Description

Save the variance explained plot as a publication-quality figure.

Usage

```
save_variance_explained_plot(
  object,
  filename = tempfile(pattern = "variance_explained_", fileext = ".tiff"),
  width = 7,
  height = 5,
  dpi = 600
)
```

Arguments

object	Object of class epi_efa.
filename	Output filename.
width	Width (inches).
height	Height (inches).
dpi	Resolution.

Value

Invisibly returns the filename.

---

scree_plot *Scree Plot*

Description

Draw a scree plot for exploratory factor analysis or Kaiser criterion results.

Usage

```
scree_plot(object)

scree_plot(object)
```

Arguments

object	An object of class epi_efa or epi_kaiser.

Value

For an epi_kaiser object, a ggplot object. For an epi_efa object, the plot is drawn using base graphics.

`sensitivity_analysis` *Sensitivity Analysis for Parallel Analysis*

Description

Repeats Horn's parallel analysis across alternative simulation counts, percentile thresholds, or analysis specifications.

Usage

```
sensitivity_analysis(
  data,
  percentiles = c(0.9, 0.95, 0.99),
  n_iters = c(100, 500, 1000),
  analyses = c("pca", "fa"),
  correlation = "auto",
  seed = 123,
  n_cores = 1L,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	A data frame, matrix, or correlation matrix.
`percentiles`	Numeric vector of percentile thresholds.
`n_iters`	Integer vector of simulation counts.
`analyses`	Character vector containing `"pca"` and/or `"fa"`.
`correlation`	Character. Correlation method.
`seed`	Integer or NULL.
`n_cores`	Integer. Number of CPU cores.
`verbose`	Logical.
`...`	Additional arguments passed to `parallel_analysis()`.

Value

A data frame of sensitivity-analysis results with class `"epi_parallel_sensitivity"`.

simulate_parallel_data *Simulate Example Questionnaire Data*

Description

Generates a reproducible synthetic questionnaire dataset with a latent multidimensional correlation structure. The resulting data can be used to demonstrate parallel analysis and related dimensionality functions in EpiQuestionR.

Usage

```
simulate_parallel_data(
  n = 300,
  n_items = 12,
  n_factors = 3,
  loading = 0.7,
  seed = NULL,
  ordinal = FALSE,
  categories = 5
)
```

Arguments

n	Number of respondents.
n_items	Number of questionnaire items.
n_factors	Number of latent factors used to generate the data.
loading	Numeric value controlling the approximate primary factor loading.
seed	Optional integer random seed.
ordinal	Logical. If TRUE, continuous responses are converted to ordinal Likert-style responses.
categories	Number of ordinal response categories when ordinal = TRUE.

Details

This function is intended for examples, demonstrations, tutorials, and package testing. It should not be used as a substitute for real data in substantive analyses.

Value

A data frame with n rows and n_items columns.

Examples

```
questionnaire_data <- simulate_parallel_data(
    n = 300,
    n_items = 12,
    n_factors = 3,
    seed = 123
)

head(questionnaire_data)
```

`summary.epi_efa` *Summary of EFA*

Description

Summary of EFA

Usage

```
## S3 method for class 'epi_efa'
summary(object, digits = 3, cutoff = 0.3, ...)
```

Arguments

`object`	epi_efa object.
`digits`	Number of decimals.
`cutoff`	Loading cutoff.
`...`	Additional arguments passed to the underlying function or method.

Value

Prints detailed summary.

`summary.epi_kaiser` *Summarize a Kaiser Criterion Analysis*

Description

Summarize a Kaiser Criterion Analysis

Usage

```
## S3 method for class 'epi_kaiser'
summary(object, digits = object$digits, ...)
```

Arguments

object	An object of class epi_kaiser.
digits	Number of decimal places to display.
...	Additional arguments, currently unused.

Value

An object of class summary.epi_kaiser.

---

summary.epi_map *Summarize a Velicer MAP Test*

Description

Creates a detailed summary of an "epi_map" object produced by map_test().

Usage

```
## S3 method for class 'epi_map'
summary(object, ...)
```

Arguments

object	An object of class "epi_map".
...	Additional arguments currently ignored.

Value

An object of class "summary.epi_map" containing the MAP recommendation, MAP criterion values, eigenvalue information, settings, metadata, and original function call.

Examples

```
set.seed(123)

dat <- data.frame(
  item1 = rnorm(300),
  item2 = rnorm(300),
  item3 = rnorm(300),
  item4 = rnorm(300),
  item5 = rnorm(300)
)

fit <- map_test(dat)
summary(fit)
```

`summary.epi_parallel` *Summarize an EpiQuestionR Parallel Analysis*

Description

Produces a structured summary of a Horn's Parallel Analysis result.

Usage

```
## S3 method for class 'epi_parallel'
summary(object, ...)
```

Arguments

`object`	An object of class `"epi_parallel"`.
`...`	Additional arguments passed to or from other methods.

Details

The returned summary includes analysis settings, observed eigenvalues, simulated reference eigenvalues, retention decisions, and optional variance information.

Value

An object of class `"summary_epi_parallel"` containing:

call The original function call.

settings Parallel analysis settings.

n_variables Number of variables analyzed.

n_observations Number of observations, when available.

n_retained Recommended number of retained components or factors.

retained_indices Indices of retained components or factors.

results Component-level parallel analysis results.

variance Optional variance-explained table.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)
summary(result)
```

tidy.epi_kaiser *Tidy Kaiser Criterion Results*

Description

Convert a `epi_kaiser` object into a tidy tibble.

Usage

```
## S3 method for class 'epi_kaiser'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An object of class `epi_kaiser`.
`conf.int`	Ignored.
`conf.level`	Ignored.
`...`	Additional arguments passed to the underlying function or method.

Value

A tibble with one row per component.

tidy.epi_map *Tidy a Velicer MAP Test*

Description

Converts the component-level results of a Velicer Minimum Average Partial (MAP) test into a tidy data frame.

Usage

```
## S3 method for class 'epi_map'
tidy(x, ...)
```

Arguments

`x`	An object of class `"epi_map"`.
`...`	Additional arguments currently ignored.

Value

A data frame with one row for each number of partialled components. The output contains the component number, MAP criterion value, minimum indicator, recommendation indicator, and criterion used.

Examples

```
set.seed(123)
dat <- matrix(rnorm(1000), ncol = 10)
fit <- map_test(dat)
generics::tidy(fit)
```

`tidy.epi_parallel` *Tidy an EpiQuestionR Parallel Analysis*

Description

Converts component- or factor-level results from Horn's Parallel Analysis into a tidy data frame.

Usage

```
## S3 method for class 'epi_parallel'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An object of class `"epi_parallel"`.
`conf.int`	Logical. Included for compatibility with common **broom**-style workflows. Confidence intervals are not defined for standard parallel-analysis eigenvalue output and are therefore not added to the returned table.
`conf.level`	Numeric confidence level. Included for interface compatibility. Must be between 0 and 1.
`...`	Additional arguments. Currently unused.

Details

Each row represents one component or factor. The output includes observed eigenvalues, simulated mean eigenvalues, simulated percentile eigenvalues, the reference eigenvalue used for retention, the difference between the observed and reference eigenvalues, and the retention decision.

Value

A data frame with one row per component or factor and columns:

component Component or factor index.

observed Observed eigenvalue.

simulated_mean Mean simulated eigenvalue.

simulated_percentile Percentile-based simulated eigenvalue.

reference Reference eigenvalue used for the retention decision.

difference Observed eigenvalue minus the reference eigenvalue.

retained Logical retention indicator.

decision Human-readable retention decision.

Examples

```
set.seed(123)
dat <- data.frame(
  item1 = rnorm(30),
  item2 = rnorm(30),
  item3 = rnorm(30),
  item4 = rnorm(30),
  item5 = rnorm(30)
)
result <- parallel_analysis(dat, n_iter = 100)

generics::tidy(result)
```

---

tidy.summary.epi_map *Tidy a Summary of a Velicer MAP Test*

Description

Converts an object returned by `summary.epi_map()` into a tidy component-level data frame.

Usage

```
## S3 method for class 'summary.epi_map'
tidy(x, ...)
```

Arguments

`x`	An object of class `"summary.epi_map"`.
`...`	Additional arguments currently ignored.

Value

A tidy data frame.

---

tli *Tucker Lewis Index*

Description

Tucker Lewis Index

Usage

```
tli(object)
```

Arguments

object	epi_efa object.

Value

Numeric.

---

uniqueness *Uniqueness*

Description

Uniqueness

Usage

```
uniqueness(object)
```

Arguments

object	epi_efa object.

Value

Numeric vector.

---

uniqueness_plot *Uniqueness Plot*

Description

Draw a publication-quality bar chart of item uniqueness values.

Usage

```
uniqueness_plot(
  object,
  sort = TRUE,
  horizontal = TRUE,
  fill = "#D95F02",
  title = "Uniqueness",
  digits = 2
)
```

Arguments

object	Object of class epi_efa.
sort	Logical. Sort uniqueness values from highest to lowest?
horizontal	Logical. Draw horizontal bars?
fill	Fill colour.
title	Plot title.
digits	Number of decimal places for labels.

Details

Uniqueness represents the proportion of variance not explained by the extracted common factors (Uniqueness = 1 - Communality).

Value

A ggplot object.

Examples

```
set.seed(123)

latent <- rnorm(200)

example_data <- data.frame(
  Q1 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q2 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q3 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q4 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q5 = 0.6 * latent + rnorm(200, sd = 0.5)
)

fit <- efa(
  example_data,
  items = paste0("Q", 1:5),
  nfactors = 1
)

uniqueness_plot(fit)
```

---

variance_explained *Variance Explained*

Description

Variance Explained

Usage

```
variance_explained(x, ...)
```

Arguments

x	An object of class epi_kaiser.
...	Additional arguments passed to other methods or currently unused.

Value

A numeric vector containing the proportion of total variance explained by each component or factor in the epi_kaiser object. Each value represents the relative contribution of the corresponding component or factor to the total variance and can be used to assess its importance.

---

variance_explained.epi_efa

Variance Explained for EFA

Description

Extracts the variance explained from an exploratory factor analysis object.

Usage

```
## S3 method for class 'epi_efa'
variance_explained(x, ...)
```

Arguments

x	An object of class epi_efa.
...	Additional arguments, currently unused.

Value

A data frame containing variance explained statistics.

variance_explained_plot *Variance Explained Plot*

Description

Draw a publication-quality plot showing the percentage of variance explained by each extracted factor and the cumulative percentage of variance explained.

Usage

```
variance_explained_plot(
  object,
  cumulative = TRUE,
  bar_fill = "#4C78A8",
  line_colour = "#D62728",
  point_colour = "#D62728",
  title = "Variance Explained"
)
```

Arguments

object	Object of class epi_efa.
cumulative	Logical. Display cumulative variance line?
bar_fill	Fill colour for bars.
line_colour	Colour for cumulative variance line.
point_colour	Colour for cumulative variance points.
title	Plot title.

Value

A ggplot object.

Examples

```
set.seed(123)

latent <- rnorm(200)

example_data <- data.frame(
  Q1 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q2 = 0.8 * latent + rnorm(200, sd = 0.5),
  Q3 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q4 = 0.7 * latent + rnorm(200, sd = 0.5),
  Q5 = 0.6 * latent + rnorm(200, sd = 0.5)
)
```

```
fit <- efa(
  example_data,
  items = paste0("Q", 1:5),
  nfactors = 1
)

variance_explained_plot(fit)
```

`variance_table` *Variance Explained Table*

Description

Variance Explained Table

Usage

```
variance_table(object)
```

Arguments

`object`	epi_kaiser object.

Value

data.frame

Index