Package 'stratifyR'

November 13, 2026

Type Package

Title Optimal Stratification of Univariate Populations

Version 2.1-0

Date 2026-11-12

Description Determines Optimum Strata Boundaries (OSB) and Optimum Sample Sizes (OSS) for univariate stratified sampling designs under Neyman allocation. The stratification variable is described by a best-fitting parametric distribution, selected automatically by AIC from a set of continuous families (normal, log-normal, gamma, Weibull, exponential, Cauchy, uniform, Pareto, triangular and right-triangular), and the optimum boundaries are obtained by minimising the Neyman objective. Version 2.0 keeps the Dynamic Programming (DP) solver of Reddy and Khan (2020), which is globally optimal for the fitted distribution, as the default. It adds two fast derivative-free solvers, a multi-start 'COBYLA' solver and a two-phase global solver that couples 'DIRECT-L' with 'BOBYQA' refinement, and an exact solver that runs dynamic programming on the sorted data and returns the partition with the smallest empirical Neyman cost. It also provides cost-constrained allocation with unequal per-stratum costs, a design-efficiency comparison (compare_designs), two- and three-dimensional and interactive visualisations, solution-quality diagnostics (a Cauchy-Schwarz optimality gap and KKT first-order residuals for the derivative-free solvers), accessor functions for the results and a self-contained 'shiny' application, while remaining backward compatible with the strata.data() and strata.distr() interface of version 1.x. The methodology follows Khan et al. (2008) <https://www150.statcan.gc.ca/n1/pub/12-001-x/2008002/article/10761-eng.pdf>, Reddy and Khan (2018) <doi:10.1111/anzs.12244> and Reddy and Khan (2020) <doi:10.1111/anzs.12301>.

License GPL (>= 3)

URL https://github.com/karunaredz/stratifyR

BugReports https://github.com/karunaredz/stratifyR/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports stats, utils, graphics, grDevices, MASS, fitdistrplus (>= 1.1.0), nloptr (>= 2.0.0), actuar, mc2d

Suggests shiny, bslib, DT, readxl, stratification, plotly (>= 4.10.0), testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, crayon, kableExtra, triangle

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

Author Karuna G. Reddy [aut, cre] (ORCID: <https://orcid.org/0000-0002-9467-4668>), M. G. M. Khan [aut]

Maintainer Karuna G. Reddy <karuna.reddy@auckland.ac.nz>

Repository CRAN

Date/Publication 2026-11-13 07:30:02 UTC

Contents

accessors . . . 2
anaemia . . . 4
compare_designs . . . 5
hies . . . 7
math . . . 7
plot.stratifyR . . . 8
print.stratifyR . . . 9
strata.data . . . 10
strata.distr . . . 12
stratifyRApp . . . 15
sugarcane . . . 16
summary.stratifyR . . . 17

Index **19**

`accessors` *Get the results of a stratification*

Description

These functions return the main results of `strata.data`, `strata.distr` and `compare_designs`, so that the internal structure of the returned objects need not be used.

Usage

```
boundaries(x, ...)

## S3 method for class 'stratifyR'
boundaries(x, ends = FALSE, ...)

allocation(x, ...)

## S3 method for class 'stratifyR'
allocation(x, ...)

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

objective(x, ...)

## S3 method for class 'stratifyR'
objective(x, ...)

## S3 method for class 'compare_designs'
objective(x, ...)

variance(x, ...)

## S3 method for class 'stratifyR'
variance(x, ...)

fitted_distribution(x, ...)

## S3 method for class 'stratifyR'
fitted_distribution(x, ...)

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

Arguments

`x`	An object of class `"stratifyR"`, or for `objective()` and `as.data.frame()` also of class `"compare_designs"`.
`...`	Not used.
`ends`	Logical. If `TRUE`, add the smallest and largest values of the variable to the boundaries. Default `FALSE`.
`row.names, optional`	Not used; for compatibility with the generic.

Details

`boundaries()` returns the optimum strata boundaries (OSB): the $L - 1$ cut points between the L

strata, or, with `ends = TRUE`, the $L + 1$ values from the smallest to the largest value of the variable.

`allocation()` returns one row per stratum, with the lower and upper boundary, the population size N_h, the weight $W_h = N_h/N$, the standard deviation S_h, the product $W_h S_h$, the sample size n_h and the sampling fraction $f_h = n_h/N_h$. `as.data.frame()` on a `"stratifyR"` object gives the same table.

`objective()` returns the value of the objective function $\Phi = \sum_h W_h S_h$ at the boundaries, the quantity that the solvers minimise.

`variance()` returns the variance of the estimated population mean under the design, with the finite population correction: $V = \sum_h W_h^2 S_h^2 (1/n_h - 1/N_h)$, where a stratum that is sampled completely adds nothing.

`fitted_distribution()` returns the name of the best-fit distribution and its parameter estimates on the scale of the data.

For a `"compare_designs"` object, `as.data.frame()` returns one row per design (SRS, proportional and Neyman allocation), with the variance and standard error of the sample mean and the design effect, and `objective()` returns Φ.

Value

`boundaries()`, `objective()` and `variance()` return numeric values, `allocation()` and `as.data.frame()` return data frames, and `fitted_distribution()` returns a list with components `distr` and `estimate`.

See Also

`strata.data`, `summary.stratifyR`, `compare_designs`

Examples

```
data(sugarcane)
res <- strata.data(sugarcane$Production, h = 4, n = 500, method = "exact")
boundaries(res)
allocation(res)
objective(res)
variance(res)
fitted_distribution(res)
as.data.frame(compare_designs(res))
```

---

anaemia *Micronutrient data on anaemia in Fiji*

Description

The Anaemia data comes from the Fiji National Nutritional Survey in 2004 on the "Micronutrient Status of Women in Fiji".

Usage

```
data(anaemia)
```

Format

A population data frame with 724 rows on some of the key components collected in the survey. The variables are:

`Haemoglobin` Level of Haemoglobin (mmol/L)

`Iron` Level of Iron (ng/mL)

`Folate` Level of Folate (mmol/L)

Source

This survey was conducted by the Ministry of Heath in Fiji. More details can be found at: https://ghdx.healthdata.org/record/fiji-national-nutrition-survey-2004

Examples

```
data(anaemia)
head(anaemia)
Iron <- anaemia$Iron
min(Iron); max(Iron)
hist(anaemia$Haemoglobin)
boxplot(anaemia$Folate)
```

---

compare_designs *Compare survey design efficiencies*

Description

Given a fitted `"stratifyR"` object (from `strata.data` or `strata.distr`), computes and compares the variance of the sample mean under three designs for the **same** total sample size n:

SRS Simple random sampling without replacement (baseline).

Proportional Stratified sampling with $n_h \propto W_h$ (proportional allocation).

Neyman Stratified sampling with $n_h \propto W_h S_h$ (optimal/Neyman allocation), the allocation used by `stratifyR`.

Usage

```
compare_designs(object, ...)

## Default S3 method:
compare_designs(object, ...)

## S3 method for class 'stratifyR'
compare_designs(object, n = NULL, ...)

## S3 method for class 'compare_designs'
print(x, digits = 6, ...)
```

Arguments

`object`	An object of class `"stratifyR"`.
`...`	Currently unused.
`n`	Integer. Total sample size. Defaults to `object$nhTot` (the value used when the stratification was run).
`x`	A `"compare_designs"` object.
`digits`	Integer. Number of significant digits used when printing.

Details

Design effects (DEFF) follow Kish (1965): $\mathrm{DEFF} = V_{\mathrm{design}}/V_{\mathrm{SRS}}$. The *equivalent SRS sample size* is the number of observations an SRS design would need to match the precision of the Neyman design, and the *cost saving* is the corresponding percentage reduction.

Value

An object of class `"compare_designs"` (an invisibly-printed list) with components:

`n` Total sample size used.

`H` Number of strata.

`S2` Estimated population variance S^2.

`V_within` Within-stratum variance component $\sum W_h S_h^2$.

`V_srs`, `V_prop`, `V_opt` Variance of $\bar{y}$ under SRS, proportional, and Neyman designs.

`SE_srs`, `SE_prop`, `SE_opt` Corresponding standard errors.

`deff_prop`, `deff_opt` Design effects relative to SRS.

`n_srs_equiv` Equivalent SRS sample size for same precision as the Neyman design.

`pct_saving` Percentage sample-size saving of Neyman over SRS.

`WhShTot` $\sum W_h S_h$ (objective function value).

See Also

`strata.data`, `strata.distr`

Examples

```
## Not run:
res <- strata.data(data = anaemia$Iron, h = 3, n = 300)
cd  <- compare_designs(res)
cd

## End(Not run)
```

hies *Household Income and Expenditure Survey (HIES) data from Fiji*

Description

The hies data comes from the HIES survey conducted in Fiji in the year 2010. The data contains only two aspects of the survey.

Usage

```
data(hies)
```

Format

A data frame with 3566 observations on two of the major quantities collected in the survey. The variables are:

`Expenditure` Level of expenditure (FJD)

`Income` Level of income (FJD)

Source

This survey was conducted in 2010 by the Bureau of Statistics (FIBoS) - Fiji Government.

Examples

```
data(hies$Income)
min(hies$Income); max(hies$Income)
hist(hies$Income)
boxplot(hies$Income)
```

math *Mathematics marks for first-year university students*

Description

The data contains the mathematics coursework marks, final examination marks and grades obtained by students in a first year mathematics course at The University level in the year 2010 in Fiji.

Usage

```
data(math)
```

Format

A data frame with 353 observations which represent mathematics marks and grades for first year math students at university level. The variable is as follows:

`cw` Coursework marks in 1st year mathematics (0-50)

`end_exam` The end of semester examination marks maths (0-50)

`final_marks` Final examination marks in maths, which is an addition of the cw and end_exam (0-100)

`grade` The grade obtained by the student based on the final marks

Source

The data was obtained by a masters students at USP, Fiji.

Examples

```
data(math)
min(math$final_marks); max(math$final_marks)
hist(math$final_marks)
boxplot(math$final_marks)
```

`plot.stratifyR` *Plot a stratification*

Description

Visualises a stratified design. The `"2d"` mode draws a histogram with the fitted density overlaid and the strata shaded; `"3d"` and `"interactive"` render **plotly** views of the Neyman cost surface.

Usage

```
## S3 method for class 'stratifyR'
plot(
  x,
  type = c("2d", "3d", "interactive"),
  data = NULL,
  n_pts = 512L,
  alpha = 0.25,
  palette = c("#4E79A7", "#F28E2B", "#E15759", "#76B7B2", "#59A14F", "#EDC948",
    "#B07AA1", "#FF9DA7", "#9C755F"),
  main = NULL,
  show_info = TRUE,
  ...
)
```

Arguments

x	A "stratifyR" object.
type	"2d" (default), "3d", or "interactive".
data	Optional numeric vector of population values.
n_pts	Integer. Density grid size. Default 512L.
alpha	Numeric. Stratum fill transparency. Default 0.25.
palette	Character vector of stratum colours (recycled).
main	Character. Plot title (auto-generated if NULL).
show_info	Logical. Overlay a per-stratum information box on the "2d" plot. Default TRUE.
...	Passed to hist() or plotly::layout().

Value

Invisibly returns x for "2d"; a **plotly** widget for "3d" and "interactive".

Examples

```
## Not run:
set.seed(1); y <- rgamma(2000, shape = 2, rate = 0.5)
res <- strata.data(y, h = 4, n = 400)
plot(res)
plot(res, type = "3d")
plot(res, type = "interactive")

## End(Not run)
```

---

print.stratifyR *Print a stratification*

Description

Prints a compact console summary of the stratification results. For the full per-stratum table use summary(x).

Usage

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

Arguments

x	A "stratifyR" object.
...	Currently unused.

Value

Invisibly returns x.

---

strata.data *Stratify a univariate population using the data*

Description

This function takes in the univariate population data (argument data) and a fixed sample size (n) to compute the optimum stratum boundaries (OSB) for a given number of strata (L), optimum sample sizes (nh), etc. directly from the data. The main idea used is from Khan et al (2008) whereby the problem of stratification is formulated into a Mathematical Programming Problem (MPP) using the best-fit frequency distribution and its parameters estimated from the data. This MPP is then solved for the OSB using a Dynamic Programming (DP) solution procedure.

Usage

```
strata.data(
  data,
  h,
  n,
  cost = FALSE,
  ch = NULL,
  method = c("dp", "cobyla", "global", "exact"),
  n_starts = 20L,
  max_iter = 2000L,
  tol = 1e-09,
  verbose = FALSE,
  fpc = FALSE,
  takeall = FALSE
)
```

Arguments

data	A numeric vector with the values of the stratification variable for every unit of the population. It must have no missing or infinite values and at least 2 * h values.
h	The number of strata, a whole number of at least 2.
n	The total sample size, a whole number between h and the population size.
cost	A logical: has default cost=FALSE. If it is a stratum-cost problem, cost=TRUE, with which, one must provide the Ch parameter.
ch	A numeric: denotes a vector of stratum costs. When cost=FALSE, it has a default of NULL.

method	Character. Optimisation method. "dp" (default) is the original Dynamic Programming solver, globally optimal on a 1,000-point grid for the fitted distribution. "exact" runs Dynamic Programming on the data and returns the partition with the smallest empirical Neyman cost over all partitions with at least two units per stratum; with more than 20,000 distinct cut points it searches 20,000 cut points spread evenly in rank (set options(stratifyR.exact_max_cand =) to change this). "cobyla" is a fast multi-start search (COBYLA and BOBYQA) on the empirical objective, and "global" is a DIRECT-L global search followed by local refinement.
n_starts	Integer. Number of random restarts for the COBYLA solver. Default 20L. Used only by "cobyla".
max_iter	Integer. Maximum function evaluations per COBYLA start. Default 2000L.
tol	Numeric. Convergence tolerance for COBYLA. Default 1e-9.
verbose	Logical. Print per-start progress. Default FALSE.
fpc	Logical. With method = "exact" only. If TRUE, the solver minimises the full variance of the stratified mean under Neyman allocation, including the finite population correction, $V = (\sum W_h S_h)^2/n - \sum W_h S_h^2/N$, instead of $\sum W_h S_h$. The minimum is certified when the reported optimality_gap is 0. Default FALSE.
takeall	Logical. With method = "exact" only. If TRUE, the top stratum may be made a take-all stratum, sampled completely, when this gives a lower variance; the rest of the sample is allocated by Neyman allocation over the other strata. Implies fpc = TRUE. Default FALSE.

Value

strata.data returns an object of class "stratifyR" with the Optimum Strata Boundaries (OSB; the upper boundary of each stratum, the last being the largest data value), stratum weights (Wh), stratum variances (Vh), Optimum Sample Sizes (nh), stratum population sizes (Nh) and sampling fraction (fh). It also contains variance, the variance of the stratified mean with the finite population correction for the final allocation, takeall, the number of units in a take-all top stratum (0 if there is none), and fpc, which records whether the full variance was minimised. Use boundaries, allocation, objective and variance to get the results, and summary() for the full table.

Author(s)

Karuna Reddy <karuna.reddy@auckland.ac.nz>
MGM Khan <mgm.khan@usp.ac.fj>

See Also

strata.distr, boundaries, summary.stratifyR, compare_designs, plot.stratifyR

Examples

```
data(sugarcane)
dp    <- strata.data(sugarcane$Production, h = 4, n = 500)
```

```
exact <- strata.data(sugarcane$Production, h = 4, n = 500, method = "exact")
c(dp = dp$WhShTot, exact = exact$WhShTot)

# minimise the full variance, allowing a take-all top stratum
set.seed(1)
x  <- (runif(2000)^(-1/2) - 1)          # heavy-tailed (Pareto II) population
ta <- strata.data(x, h = 4, n = 200, method = "exact", takeall = TRUE)
c(variance = ta$variance, takeall = ta$takeall)

## Not run:
data <- rweibull(1000, shape=2, scale = 1.5)
hist(data)
obj <- strata.data(data, h = 2, n=300)
summary(obj)
#-------------------------------------------------------------
data(anaemia)
Iron <- anaemia$Iron
res <- strata.data(Iron, h = 2, n=350)
summary(res)
#-------------------------------------------------------------
data(SHS) #Household Spending data from stratification package
weight <- SHS$WEIGHT
hist(weight); length(weight)
res <- strata.data(weight, h = 2, n=500)
summary(res)
#-------------------------------------------------------------
data(sugarcane)
Production <- sugarcane$Production
hist(Production)
res <- strata.data(Production, h = 2, n=1000)
summary(res)
#-------------------------------------------------------------
#The function be dynamically used to visualize the the strata boundaries,
#for 2 strata, over the density (or observations) of the "mag" variable
#from the quakes data (with purrr and ggplot2 packages loaded).
output <- quakes %>%
          pluck("mag") %>%
          strata.data(h = 2, n = 300)
quakes %>%
      ggplot(aes(x = mag)) +
      geom_density(fill = "blue", colour = "black", alpha = 0.3) +
      geom_vline(xintercept = output$OSB, linetype = "dotted", color = "red")
#-------------------------------------------------------------

## End(Not run)
```

`strata.distr` *Stratify a univariate population using a distribution*

Description

This function takes in the underlying hypothetical distribution and its parameter(s) of the survey variable, the initial value and the range of the population, the fixed sample size (n) and the fixed population size (N) to compute the optimum stratum boundaries (OSB) for a given number of strata (L), optimum sample sizes (nh), etc. The main idea used is from Khan et al. (2008) whereby the problem of stratification is fromulated into a Mathematical Programming Problem (MPP) using the best-fit frequency distribution and its parameter estimates of the data. This MPP is then solved for the optimal solutions using the Dynamic Programming (DP) solution procedure.

Usage

```
strata.distr(
  h,
  initval,
  dist,
 distr = c("pareto", "triangle", "rtriangle", "weibull", "gamma", "exp", "unif", "norm",
    "lnorm", "cauchy"),
 params = c(shape = 0, scale = 0, rate = 0, gamma = 0, location = 0, mean = 0, sd = 0,
    meanlog = 0, sdlog = 0, min = 0, max = 0, mode = 0),
  n,
  N,
  cost = FALSE,
  ch = NULL,
  method = c("dp", "cobyla", "global"),
  n_starts = 20L,
  max_iter = 2000L,
  tol = 1e-09,
  verbose = FALSE
)
```

Arguments

h	The number of strata, a whole number of at least 2.
initval	The smallest value of the stratification variable.
dist	The range of the variable (largest minus smallest value), a positive number.
distr	A character: denotes the name of the distribution that characterizes the population.
params	A named numeric vector (or list) with the parameters of the distribution: shape and scale for "pareto" and "weibull"; shape and rate for "gamma"; rate for "exp"; min and max for "unif" and "rtriangle", with mode also for "triangle"; mean and sd for "norm"; meanlog and sdlog for "lnorm"; and location and scale for "cauchy".
n	The total sample size, a whole number between h and N.
N	The population size.
cost	A logical: has default cost=FALSE. If it is a stratum-cost problem, cost=TRUE, with which one must provide the Ch parameter.

ch	A numeric: denotes a vector of stratum costs.
method	Character. Optimisation method: "cobyla" (fast multi-start COBYLA, default), "dp" (original Dynamic Programming solver), or "global" (exhaustive global search).
n_starts	Integer. Number of random restarts for the COBYLA solver. Default 20L. Ignored when method = "dp".
max_iter	Integer. Maximum function evaluations per COBYLA start. Default 2000L.
tol	Numeric. Convergence tolerance for COBYLA. Default 1e-9.
verbose	Logical. Print per-start progress. Default FALSE.

Value

strata.distr returns Optimum Strata Boundaries (OSB), stratum weights (Wh), stratum costs (Ch), stratum variances (Vh), Optimum Sample Sizes (nh), stratum population sizes (Nh), as an object of class "stratifyR". Use boundaries, allocation, objective and variance to get the results.

Author(s)

Karuna Reddy <karuna.reddy@auckland.ac.nz>
MGM Khan <mgm.khan@usp.ac.fj>

See Also

strata.data, boundaries, summary.stratifyR, compare_designs

Examples

```
## Not run:
#Assume data has initial value of 1.5, distance of 33 and follows
#weibull distribution with estimated parameters as shape=2.15 and scale=13.5
#To compute the OSB, OSS, etc. with fixed sample n=500, we use:
res <- strata.distr(h=2, initval=1.5, dist=33, distr = "weibull",
params = c(shape=2.15, scale=13.5), n=500, N=2000, cost=FALSE)
summary(res)
#-------------------------------------------------------------
#Assume data has initial value of 1, distance of 10415 and follows
#lnorm distribution with estimated parameters as meanlog=5.5 and sdlog=1.5
#To compute the OSB, OSS, etc. with fixed sample n=500, we use:
res <- strata.distr(h=2, initval=1, dist=10415, distr = "lnorm",
params = c(meanlog=5.5, sdlog=1.5), n=500, N=12000)
summary(res)
#-------------------------------------------------------------
#Assume data has initial value of 2, distance of 68 and follows
#gamma distribution with estimated parameters as shape=3.8 and rate=0.55
#To compute the OSB, OSS, etc. with fixed sample n=500, we use:
res <- strata.distr(h=2, initval=0.65, dist=68, distr = "gamma",
params = c(shape=3.8, rate=0.55), n=500, N=10000)
summary(res)
#-------------------------------------------------------------
```

```
#The function be dynamically used to visualize the the strata boundaries,
#for 2 strata, over the density (or observations) of the "mag" variable
#from the quakes data (with purrr and ggplot2 packages loaded).
res <- strata.distr(h=2, initval=4, dist=2.4, distr = "lnorm",
          params = c(meanlog=1.52681032, sdlog=0.08503554), n=300, N=1000)
quakes %>%
      ggplot(aes(x = mag)) +
      geom_density(fill = "blue", colour = "black", alpha = 0.3) +
      geom_vline(xintercept = res$OSB, linetype = "dotted", color = "red")
#-------------------------------------------------------------

## End(Not run)
```

`stratifyRApp` *Launch the stratifyR Shiny application*

Description

Opens the **stratifyR** web interface in the default browser. The application gives no-code access to the package: users upload their own data (CSV or Excel) or select a built-in dataset, compute optimum stratum boundaries and sample sizes with the `"dp"`, `"cobyla"`, `"global"` or `"exact"` solvers, compare design efficiencies, explore boundaries interactively, and download the results.

Usage

```
stratifyRApp(...)
```

Arguments

`...`	Additional arguments passed to `runApp`, for example `port` or `launch.browser`.

Details

The application needs **shiny**, **bslib** and **DT** to start; if any of these is missing, `stratifyRApp()` stops with a short installation hint rather than a cryptic error. The optional packages **plotly**, **readxl**, **ggplot2** and **stratification** add features (interactive 2D/3D and slider plots, Excel upload, and the Lavallee-Hidiroglou comparison). The app opens without them, and on launch it reports any that are not installed so nothing fails silently. To install everything the app can use:

```
install.packages(c("shiny",
"bslib", "DT", "plotly", "readxl", "ggplot2", "stratification"))
```

Value

Called for its side effect (launches a Shiny app); returns `NULL` invisibly.

Examples

```
## Not run:
stratifyRApp()

## End(Not run)
```

`sugarcane` *Sugarcane farming data from Fiji*

Description

The sugarcane data shows the disposition area (land area under cane) for individual sugarcane farms and their cane productions with the incomes/earnings for the year 2010 in Fiji.

Usage

```
data(sugarcane)
```

Format

A data frame with 13894 observations corresponding to individual farms. The following are the variables:

`DispArea` Disposition area (or land area under cane) (hactares)

`Production` The amount of sugarcane produced in the farm (tonnes)

`Income` Net income or money paid to farmers) (in FJD)

Source

This data was obtained from the Fiji Sugar Corporation in Fiji.

Examples

```
data(sugarcane$Production)
head(sugarcane$Production)
Production <- sugarcane$Production
min(Production); max(Production)
hist(Production)
boxplot(Production)
```

`summary.stratifyR` *Summarise a stratification*

Description

Builds a summary of a `"stratifyR"` object: the design settings, the fitted distribution and a table with one row per stratum and a total row. `summary()` prints nothing itself; the result is printed by its `print()` method (or automatically at the console), and `as.data.frame()` returns the table.

Usage

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

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

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

Arguments

`object`	An object of class `"stratifyR"`, from `strata.data` or `strata.distr`.
`...`	Not used.
`x`	An object of class `"summary.stratifyR"`.
`row.names`, `optional`	Not used; for compatibility with the generic.

Details

In the console, the boundary and $W_h S_h$ columns are shown in purple and the total row in bold gold when the console supports colour (through the **crayon** package; set `options(stratifyR.color = FALSE)` to turn this off). In R Markdown, the table is shown as a styled HTML table.

Value

`summary()` returns an object of class `"summary.stratifyR"`, a list with components:

`L` Number of strata.

`n`, `N` Sample size and population size.

`method` The solver used.

`distr`, `estimate` The best-fit distribution and its parameter estimates.

`range` The smallest and largest values of the variable.

`cost`, `ch` Whether stratum costs were used, and the costs.

`takeall`, `fpc`, `variance` The take-all stratum size, whether the finite population correction was used, and the variance of the mean (for `fpc = TRUE` or `takeall = TRUE`).

`table` A data frame with one row per stratum and a total row: the upper boundary (`OSB`), `Wh`, `Vh` (the stratum variance S_h^2), `WhSh`, `nh`, `Nh` and `fh`.

`print()` returns `x` invisibly, and `as.data.frame()` returns `x$table`.

See Also

`strata.data`, `boundaries`, `allocation`, `objective`

Examples

```
data(sugarcane)
res <- strata.data(sugarcane$Production, h = 4, n = 500, method = "exact")
s <- summary(res)
s
as.data.frame(s)
```

Index

∗ **datasets**
anaemia, 4
hies, 7
math, 7
sugarcane, 16

accessors, 2
allocation, *11*, *14*, *18*
allocation *(*accessors*)*, 2
anaemia, 4
as.data.frame.compare_designs *(*accessors*)*, 2
as.data.frame.stratifyR *(*accessors*)*, 2
as.data.frame.summary.stratifyR *(*summary.stratifyR*)*, 17

boundaries, *11*, *14*, *18*
boundaries *(*accessors*)*, 2

compare_designs, *2*, *4*, 5, *11*, *14*

fitted_distribution *(*accessors*)*, 2

hies, 7

math, 7

objective, *11*, *14*, *18*
objective *(*accessors*)*, 2

plot.stratifyR, 8, *11*
print.compare_designs *(*compare_designs*)*, 5
print.stratifyR, 9
print.summary.stratifyR *(*summary.stratifyR*)*, 17

runApp, *15*

strata.data, *2*, *4–6*, 10, *14*, *17*, *18*
strata.distr, *2*, *5*, *6*, *11*, 12, *17*
stratifyRApp, 15
sugarcane, 16
summary.stratifyR, *4*, *11*, *14*, 17

variance, *11*, *14*
variance *(*accessors*)*, 2