Package 'stratifyR'

September 10, 2026

Type Package

Title Optimal Stratification of Univariate Populations

Version 2.0-1

Date 2026-09-10

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 original globally optimal Dynamic Programming (DP) solver of Reddy and Khan (2020) as the default and adds two faster derivative-free alternatives for interactive and large-scale use: a multi-start 'COBYLA' solver and a two-phase 'global' solver that couples 'DIRECT-L' with 'COBYLA' refinement. 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) 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)

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], M. G. M. Khan [aut]

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

Repository CRAN

Date/Publication 2026-09-10 14:40:08 UTC

Contents

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 "strata" 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.

Design effects (DEFF) follow Kish (1965): $\text{DEFF} = V_{\text{design}}/V_{\text{SRS}}$. 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.

Usage

```
compare_designs(object, ...)

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

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

Arguments

object	An object of class "strata".
...	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.

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)
```

`create.mat` *To create and store calculated values of the objective function*

Description

This function creates a matrix whose rows and columns depend on the range or distance of the data and the number of strata solutions that the user is seeking to compute. The matrix stores the objective function values calculated by the algorithm only to be accessed later for the purpose of presenting the OSB.

Usage

```
create.mat(my_env)
```

Arguments

`my_env`	The environment my_env has various constants stored from earlier operations dealing with information on the data

Value

stores numerical quantities of the objective function and stores in the two matrices inside the my_env to be accessed by other functions

Author(s)

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

`data.alloc` *Allocate data To calculate the stratum sample sizes (nh) for a fixed sample size (n) directly based on the data*

Description

Allocate data To calculate the stratum sample sizes (nh) for a fixed sample size (n) directly based on the data

Usage

```
data.alloc(data, my_env)
```

Arguments

`data`	Input dataset.
`my_env`	Environment object.

Value

...

Uses OSB to compute stratum weights (Wh), sample variances (Vh from data), and Neyman allocations (nh). Builds the summary table once at the end.

`data.optim` *To implement the Dynamic Programming (DP) solution procedure on the stratification problem presented in the form of a Mathematical Programming Problem (MPP)*

Description

This function uses the Dynamic Programming (DP) solution procedure in solving the objective function for the univariate stratification problem. It calculates the objective function values using the brute-force algorithm and stores those values in the matrices and keeps a copy in my_env so that a global minimum could be obtained.

Usage

```
data.optim(k, n, incf, minYk, maxYk, isFirstRun = TRUE, my_env)
```

Arguments

k	A numeric: number of strata
n	A numeric: is the distance*1000
incf	A numeric: 10e-3 when k=1 and 10e-5 for k>=2
minYk	A numeric: index to access minimum elements in the matrix
maxYk	A numeric: index to access maximum elements in the matrix
isFirstRun	A boolean: TRUE/FALSE parameter
my_env	The environment my_env has various constants and calculations stored from earlier opeartions through various other functions

Value

returns the array filled with calculations of objective function values

Author(s)

Karuna Reddy <karuna.reddy@auckland.ac.nz>
M GM Khan <khan_mg@usp.ac.fj>

---

data.root *Calculate the objective function value for a given (d, y)*

Description

Used by the DP recurrences to evaluate the stratification objective at a specific remaining distance 'd' and first-stratum width 'y', given stratum cost 'c' and constants tucked in 'my_env.'

Usage

```
data.root(d, y, c, my_env)
```

Arguments

d	numeric: remaining distance/range on the (scaled) axis
y	numeric: first stratum width on the (scaled) axis
c	numeric: per stratum cost multiplier (Ch[k])
my_env	environment: holds distribution name, parameters and scaling

Value

numeric: sqrt(objective) or -1 if branch is infeasible/invalid

`distr.alloc` *To calculate the stratum sample sizes (nh) for a fixed sample size (n) based on the hypothetical distribution of the data*

Description

Uses OSB to compute stratum weights (Wh), variances (Vh), Neyman allocations (nh), and related totals under a *given* population distribution. Integrations are cached per stratum; the output data.frame is built once at the end (faster).

Usage

```
distr.alloc(my_env)
```

Arguments

`my_env`	Environment carrying all precomputed values and constants.

Value

Populates my_env$output, my_env$out, and totals (WhTot, NhTot, etc.)

`distr.optim` *To implement the Dynamic Programming (DP) solution procedure on the stratification problem presented in the form of a Mathematical Programming Problem (MPP)*

Description

This function uses the Dynamic Programming (DP) solution procedure in solving the objective function for the univariate stratification problem. It calculates the objective function values using the brute-force algorithm and stores those values in the matrices and keeps a copy in my_env so that a global minimum could be obtained.

Usage

```
distr.optim(k, n, incf, minYk, maxYk, isFirstRun = TRUE, my_env)
```

Arguments

`k`	A numeric: number of strata
`n`	A numeric: is the distance*1000
`incf`	A numeric: 10e-3 when k=1 and 10e-5 for k>=2
`minYk`	A numeric: index to access minimum elements in the matrix
`maxYk`	A numeric: index to access maximum elements in the matrix

isFirstRun	A boolean: TRUE/FALSE parameter
my_env	My environment my_env has various constants and calculations stored from earlier opeartions through various other functions

Value

returns the array filled with calculations of objective function values

Author(s)

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

---

distr.root *Calculate the objective function value for a given (d, y) under a hypothesized distribution (scaled-data formulation)*

Description

Used by the DP recurrences in the "distribution-known" pathway. All distribution parameters are interpreted on the "scaled" axis (initval .. initval+dist), consistent with 'strata.distr()'.

Usage

```
distr.root(d, y, c, my_env)
```

Arguments

d	numeric: remaining distance/range on the (scaled) axis
y	numeric: width of the first stratum on the (scaled) axis
c	numeric: per stratum cost multiplier (Ch[k])
my_env	environment: holds distribution name, scaled parameters, and scaling constants ('initval', 'maxval', etc.)

Value

numeric: sqrt(objective) or -1 if branch is infeasible/invalid

Author(s)

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

`erf` *To calculate the error for a normal variable*

Description

This function calculates the value of the error according to the normally distributed variable using the idea presented in Abramowitz and Stegun (2011)

Usage

```
erf(x)
```

Arguments

`x`	The data that is provided

Value

Gives the error for a normal variable

Author(s)

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

`get.dist` *Determine best-fit distribution Identify the best-fit distribution for a univariate numeric vector*

Description

Determine best-fit distribution Identify the best-fit distribution for a univariate numeric vector

Usage

```
get.dist(data, my_env)
```

Arguments

`data`	Input dataset.
`my_env`	Environment object.

Value

...

Fits several candidate distributions via MLE and selects the model with the lowest (finite) AIC. For strictly positive data, tail-sensitive families are also tried. If a triangular fit narrowly wins (Delta AIC <= 10 over a tail model, prefer the tail model (helps avoid spurious triangular wins on mildly skewed data).

Returns a list with: - distr: best model name (e.g., "gamma", "weibull", "triangle", ...) - params: named parameter vector for the best model - aic: named numeric vector of AICs for all attempted models (NA if failed) - fits_ok: named logical vector (TRUE where fit succeeded) - messages:named list of diagnostic messages per model (errors/warnings)

---

hies *Household Income Expenditure Survey (HIES) in 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)
```

minim.val *To identify the minimum value out of two given sets of values*

Description

This function is called in data.optim() or distr.optim() which basically compares and returns the smaller value out of two given sets of values.

Usage

```
minim.val(val1, val2)
```

Arguments

val1	A numeric: the first value
val2	A numeric: the second value

Value

returns the minimum value

Author(s)

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

---

mode.val *To calculate the modal value of the data*

Description

This function calculates the value of the mode of the data that is provided

Usage

```
mode.val(x)
```

Arguments

x	The data that is provided

Value

Gives the mode

Author(s)

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

plot.strata *Plot Method for Stratified Survey Design Objects*

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 'strata'
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 "strata" 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.strata` *Print Method for Stratified Survey Design Objects*

Description

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

Usage

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

Arguments

`x`	A `"strata"` object.
`...`	Currently unused.

Value

Invisibly returns `x`.

`realloc` *To re-allocate the stratum sample sizes (nh)*

Description

This function re-calculates or re-allocate the stratum sample sizes (nh) after it has already been initially allocated via Neyman allocation. This is applied to resolve the problem of oversampling in one or more of the strata.

Usage

```
realloc(h, x, nh, Nh, nume, my_env)
```

Arguments

h	A numeric: the no. of strata
x	A vector: the osb that has been calculated
nh	A vector: the stratum sample sizes that have been initially calculated
Nh	A vector: the stratum population sizes that have been initially calculated
nume	A numeric: the numerator total
my_env	The environment my_env has various constants and outputs stored from earlier opeartions through various other functions

Value

calculates and presents the new re-allocate stratum samples

Author(s)

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

---

strata.data *Stratification of Univariate Survey 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"),
            n_starts = 20L, max_iter = 2000L, tol = 1e-09,
            verbose = FALSE)
```

Arguments

data	A vector of values of the survey variable y for which the OSB are determined.
h	A numeric: denotes the number of strata to be created.
n	A numeric: denotes a fixed total sample 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: "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.data returns Optimum Strata Boundaries (OSB), stratum weights (Wh), stratum variances (Vh), Optimum Sample Sizes (nh), stratum population sizes (Nh) and sampling fraction (fh).

Author(s)

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

See Also

strata.distr

Examples

```
## 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` *Stratification of Univariate Survey Population Using the 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	A numeric: denotes the number of strata to be created.
initval	A numeric: denotes the initial value of the population.
dist	A numeric: denotes distance (or range) of the population.
distr	A character: denotes the name of the distribution that characterizes the population.
params	A list: contains the values of all parameters of the distribution.
n	A numeric: denotes the fixed total sample size.
N	A numeric: denotes the fixed total 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).

Author(s)

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

See Also

strata.data

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 2.0 Interactive 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"` or `"global"` 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 in 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.strata` *Format and Present Results*

Description

Format and Present Results

Usage

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

Arguments

`object`	A strata object.
`...`	Additional arguments.
	Nicely formatted (and colored) summary for "strata" objects Console: aligned ASCII table with crayon colors (if supported). HTML: kable + kableExtra with yellow TOTAL row (text only, no background).

Index