Package 'eVCGsampler'

November 12, 2026

Type Package

Title VCG Sampling using Energy-Based Covariate Balancing

Version 1.2.1

Description
Provides a principled framework for sampling Virtual Control Group (VCG) using energy distance-based covariate balancing. The package offers visualization tools to assess covariate balance and includes a permutation test to evaluate the statistical significance of observed deviations.

License MIT + file LICENSE

URL https://github.com/Sanofi-Public/eVCGsampler

Encoding UTF-8

RoxygenNote 7.3.3

Depends ggplot2

Imports ggforce, osqp, patchwork, shiny

Suggests bslib, DT, glue, knitr, openxlsx, readxl, rmarkdown, shinycssloaders, shinyhelper, shinyjs, testthat (>= 3.0.0)

VignetteBuilder knitr

NeedsCompilation no

Author Andreas Schulz [aut, cre] (Created in 2025), Sanofi [cph, fnd]

Maintainer Andreas Schulz <andreas.schulz2@sanofi.com>

Repository CRAN

Date/Publication 2026-11-12 15:10:02 UTC

Contents

BestVCGsize 2
combine_data 3
combine_variables 4
energy_distance 5
energy_test 6

find_outliers 7
launchApp 8
multiSampler 9
plot_var 10
robust_scale 11
VCG_sampler 12

Index **14**

`BestVCGsize` *The function attempts to find the optimal size for VCG.*

Description

The function tries out different sizes of VCG and searches for the smallest distance.

Usage

```
BestVCGsize(formula, data = data, plot = TRUE)
```

Arguments

`formula`	A formula specifying the treated and covariates, e.g., 'treated ~ cov1 + cov2 | stratum'. The treated variable must be binary (0=pool, 1=treated)
`data`	A data frame containing the variables specified in the formula.
`plot`	Logical. If 'TRUE', returns a ggplot2 plot. Default: TRUE

Details

It is only intended for exploratory purposes, as the VCG size is normally given. But it can be used to see how well the given size fits. The recommendation for VCG size is based solely on distance and does not take into account other aspects such as power or validity.

Value

If 'plot = TRUE', returns a list with:

`optimal_n`	The estimated optimal VCG size (integer).
`plot`	A ggplot2 object visualizing the energy distance curve and plateau.

Examples

```
set.seed(2342)
dat <- data.frame(
  treat = rep(0:1, c(50, 30)),
  cov1 = c(rnorm(50, 11, 2),  rnorm(30, 10, 1)),
  cov2 = c(rnorm(50, 12, 2),  rnorm(30, 10, 1)),
  cov3 = c(rnorm(50, 9,  2),  rnorm(30, 10, 1))
)
```

```
BestVCGsize(treat ~ cov1 + cov2 + cov3, data=dat)
```

`combine_data` *Combine data from pool and treated groups*

Description

If your data is stored in separate files, you can use this function to merge them.

Usage

```
combine_data(POOL_data, TG_data, indicator_name = "treated")
```

Arguments

`POOL_data`	Data frame with POOL data, where you want to sample from.
`TG_data`	Data frame with TG (treated groups) data, all treated groups together!
`indicator_name`	Name of the variable that is created for further use in the package, Default: 'treated'

Value

Data frame with all covariates that were present in both files and with new indicator factor POOL vs TG

Examples

```
pool_data <- data.frame(
  cov1   = rnorm(100, 11, 2),
  cov2   = rnorm(100, 11, 2),
  cov3   = rnorm(100, 11, 2),
  sex    = rbinom(100, 1, 0.5))

tg_data <- data.frame(
  cov2   = rnorm(20, 12, 1),
  cov3   = rnorm(20, 12, 1),
  cov4   = rnorm(20, 12, 1),
  sex    = rbinom(20, 1, 0.5))

 dx <- combine_data(pool_data, tg_data)
 str(dx)
```

`combine_variables` *Compute Weighted Combined Score from Multiple ariables*

Description

Creates a single combined variable from multiple ariables, weighted by relative importance scores. Variables are scaled before combination.

Usage

```
combine_variables(formula, data, weights = NULL)
```

Arguments

`formula`	A formula specifying the ariables, e.g., '1 ~ cov1 + cov2 + cov3'. The left side should be 1 (placeholder).
`data`	A data frame containing the variables specified in the formula.
`weights`	A numeric vector of relative importance weights for each variable. Must have the same length as the number of variables in the formula. Higher weights indicate greater contribution (correlation) to the combined score. If NULL, equal weights (all 1s) are used. Default: NULL.

Details

Variables are first scaled to mean 0 and standard deviation 1, then multiplied by their importance weights and summed. The final score is a weighted linear combination: score = w1*z1 + w2*z2 + ... + wk*zk, where zi are the scaled variables and wi are the weights.

Value

A numeric vector of the same length as nrow(data), representing the combined weighted score for each observation. The score correlates with each input variable proportionally to its importance weight.

Examples

```
dat <- data.frame(
  age    = rnorm(80, 5, 2),
  weight = rnorm(80, 11, 2),
  class  = rbinom(80, 3, 0.5)
)

# Equal weights
score <- combine_variables(1 ~ age + weight + class, data = dat)

# Custom weights: age contributes 2x, weight 1.5x, class 1x
score <- combine_variables(1 ~ age + weight + class, data = dat,
                           weights = c(2, 1.5, 1))
```

`energy_distance` *Compute Energy Distance Between Two Groups*

Description

Calculates the energy distance between two groups.

Usage

```
energy_distance(formula, data, standardized = TRUE)
```

Arguments

`formula`	A formula specifying the treated and covariates, e.g., 'treated ~ cov1 + cov2'. The treated variable must be binary (0=pool, 1=treated)
`data`	A data frame containing the variables specified in the formula.
`standardized`	If TRUE, the standardized energy distance that lies in the range 0 to 1 is returned, the so-called E-coefficient. If FALSE, non-scaled energy distance is returned that can be >1.

Details

Energy distance is a non-parametric measure of distributional difference. It is sensitive to differences in location, scale, and shape between groups. Before calculation, the covariates are scaled to a mean value of 0 and a standard deviation of 1.

Value

A numeric value representing the energy distance between the two groups.

Examples

```
dat <- data.frame(
 treated = rep(0:1, c(50, 30)),
 age    = c(rnorm(50, 5, 2),   rnorm(30, 5, 1)),
 weight = c(rnorm(50, 11, 2),  rnorm(30, 10, 1)),
 class  = c(rbinom(50, 3, 0.6),   rbinom(30, 3, 0.4))
 )

 energy_distance(treated ~ age + weight + class, data=dat)
```

`energy_test` *Permutation Energy Test for Covariate Imbalance*

Description

Performs a permutation-based energy distance test to assess whether two groups (defined by a binary treated variable) are balanced across a set of covariates. Optionally, it visualizes the distribution of permuted energy distances and highlights the observed test statistic and critical value.

Usage

```
energy_test(formula, data, alpha = 0.05, R = 2000, plot = TRUE)
```

Arguments

`formula`	A formula specifying the treated and covariates, e.g., 'treated ~ cov1 + cov2 | stratum'.
`data`	A data frame containing the variables specified in the formula.
`alpha`	Significance level for the test (default is 0.05).
`R`	Number of permutations to perform (default is 2000).
`plot`	Logical. If 'TRUE', returns a ggplot2 visualization of the permutation distribution.

Details

The energy distance is a non-parametric measure of distributional difference. This test evaluates whether the covariate distributions between two groups are statistically distinguishable. A small p-value indicates imbalance between groups. A one-sided test is used because the energy distance is strictly positive; only values greater than the observed statistic in the permutation distribution are relevant.

Value

If 'plot = TRUE', returns a list with:

- A list of class '"htest"' containing:
  - 'p.value': The permutation p-value.
  - 'estimate': The observed energy distance.
  - 'critical.value': The critical value at the specified alpha level.
  - 'alternative': The alternative hypothesis ("one.sided").
  - 'method': Description of the test.
  - 'n.permutations': Number of permutations performed.
  - 'permutations': Vector of permuted energy distances.
- A ggplot2 object showing the histogram of permuted distances, with vertical lines for the observed statistic and critical value.

If 'plot = FALSE', returns only the '"htest"' result list.

See Also

element

Examples

```
dat <- data.frame(
 treated = rep(0:1, c(50, 30)),
 age    = c(rnorm(50, 5, 2),   rnorm(30, 5, 1)),
 weight = c(rnorm(50, 11, 2),  rnorm(30, 10, 1)),
 class  = c(rbinom(50, 3, 0.6),   rbinom(30, 3, 0.4))
 )

 energy_test(treated ~ age + weight + class, data=dat, R = 500)
```

---

find_outliers *Find Outlier Groups Based on Energy Distance*

Description

Identifies groups (e.g., studies) that are most distant from the average group based on energy distance across multiple variables.

Usage

```
find_outliers(formula, data, cutoff = 0.99, R = 500, plot = TRUE)
```

Arguments

formula	A formula specifying the group variable and variables. e.g., 'study ~ var1 + var2 +...'. The group variable should be a factor or will be converted to one.
data	A data frame containing the variables specified in the formula.
cutoff	Numeric. Percentile threshold for the permutation-based cutoff (default 0.99). The cutoff is determined by permuting group labels and calculating the percentile of permuted median distances.
R	Integer. Number of permutations for determining the cutoff (default 500).
plot	Logical. If TRUE (default), returns a visualization of the outlier analysis.

Details

Groups with high median distance to other groups are identified as potential outliers. The outlier_score is a z-score that indicates how many standard deviations a group's median distance is from the overall median distance.

Before distance calculation, all covariates are scaled to mean 0 and standard deviation 1.

Value

If 'plot = TRUE', returns a list with:

- 'cutoff_value': The permutation-based cutoff value used for outlier detection.
- 'summary': Data frame with group, median_distance, outlier_score, and is_outlier columns.
- 'heatmap': A ggplot2 heatmap of pairwise energy distances.
- 'barplot': A ggplot2 bar plot showing median distance to other groups.

If 'plot = FALSE', returns only the elements without plots.

Examples

```
# Example 1: 10 studies with real outliers (Study-8, Study-9, Study-10)
set.seed(123)
dat <- data.frame(
  study = factor(rep(paste0("Study-", 1:10), each = 20)),
  var1 = c(rnorm(20, 10, 1), rnorm(20, 10, 1), rnorm(20, 10, 1), rnorm(20, 10, 1),
           rnorm(20, 10, 1), rnorm(20, 10, 1), rnorm(20, 10, 1), rnorm(20, 15, 1),
           rnorm(20, 10, 1), rnorm(20, 16, 1)),
  var2 = c(rnorm(20, 5, 1), rnorm(20, 5, 1), rnorm(20, 5, 1), rnorm(20, 5, 1),
           rnorm(20, 5, 1), rnorm(20, 5, 1), rnorm(20, 5, 1), rnorm(20, 5, 1),
           rnorm(20, 10, 1), rnorm(20, 5, 1))
)
out <- find_outliers(study ~ var1 + var2, data = dat, R = 200)
out$summary      # Study-8, Study-9, Study-10 should be flagged
out$cutoff_value # Permutation-based threshold

# Example 2: 20 studies with NO real outliers (all from same distribution)
set.seed(456)
dat_no_outliers <- data.frame(
  study = factor(rep(paste0("Study-", 1:20), each = 15)),
  var1 = rnorm(300, 10, 2),
  var2 = rnorm(300, 5, 1)
)
out2 <- find_outliers(study ~ var1 + var2, data = dat_no_outliers, R = 200)
out2$summary     # Should have few or no outliers flagged
sum(out2$is_outlier)  # Count of flagged outliers (expected: 0 or very few)
```

`launchApp` *Launch the Shiny App*

Description

Starts the interactive eVCGsampler app: upload the POOL and TG data, choose the covariates and strata, run the pre-tests, draw the VCG and download an HTML report and the sampled data.

Usage

```
launchApp(...)
```

Arguments

`...`	Further arguments passed to `runApp`, e.g. `port` or `launch.browser`.

Details

The app needs some packages that are not required by the rest of eVCGsampler (listed under Suggests). Missing ones are reported together with the command to install them. Rendering the HTML report additionally needs pandoc (it ships with RStudio).

Examples

```
## Not run:
eVCGsampler::launchApp()

## End(Not run)
```

---

multiSampler *Multi-Sample VCG Generator and Overlap Visualization*

Description

Repeatedly samples VCGs (via 'VCG_sampler' with 'random=TRUE') from the pool, optionally plots the overlap of VCGs.

Usage

```
multiSampler(formula, data, n, c_w = NULL, Nsamples = 20, plot = TRUE)
```

Arguments

`formula`	A formula specifying the treated and covariates, e.g., 'treated ~ cov1 + cov2'. The treated variable must be binary (0=pool, 1=treated)
`data`	A data frame containing the variables specified in the formula.
`n`	Integer. Number of observations to sample from the pool. Or a vector of n for each stratum.
`c_w`	Optional: Vector of positive weights for covariates, reflecting the relative importance of the covariates for the balancing.
`Nsamples`	Number of VCGs to generate (default is 20).
`plot`	Logical; if 'TRUE', returns a ggplot2 plot showing the overlap of VCGs (default is 'TRUE').

Details

The function repeatedly calls ‘VCG_sampler‘ with ‘random‘ set to TRUE to generate multiple VCGs. It calculates the frequency of selection for each observation and computes the average percentage of overlapping observations. This function should only be used if you really need multiple VCGs, e.g. for PoC studies. It is not intended for selecting one VCG from them afterwards! In this case, the VCG_sampler function should be used directly and only one VCG should be generated.

Value

If ‘plot = TRUE‘, returns a list with:

data The original data frame with additional VCG columns (‘VCG_1‘, ..., ‘VCG_Nsamples‘).

p A ‘ggplot2‘ object showing the number of times each observation was selected across VCG samples.

If ‘plot = FALSE‘, returns the modified data frame only.

Examples

```
dat <- data.frame(
treat = rep(0:1, c(50, 30)),
cov_1 = c(rnorm(50, 5, 2),   rnorm(30, 5, 1)),
cov_2 = c(rnorm(50, 11, 2),  rnorm(30, 10, 1))
)

result <- multiSampler(treat ~ cov_1 + cov_2, data = dat, n = 10, Nsamples = 10)
```

---

plot_var *Visualize Covariate Distribution Across TG, VCG, and POOL*

Description

Creates a plot to compare the distribution of a selected variable across three groups: TG (treated groups), VCG (virtual control group), and POOL (data pool).

Usage

```
plot_var(data, what = NULL, stratum = "in_stratum", group = "VCG", title = "")
```

Arguments

data	A balanced data frame (output of the VCG_sampler function)
what	A string specifying the name of the variable to be visualized.
stratum	A string specifying the name of the stratum variable (default is ‘"in_stratum"‘)
group	A string specifying the column name used to define group membership (default is ‘"VCG"‘).
title	Optional title for the plot.

Details

The function uses energy distance to quantify distributional differences between groups. For continuous variables, it overlays dashed lines for TG group statistics (mean, min, max) and displays sample sizes. For categorical variables, it uses color-coded bars and cumulative proportion lines to highlight imbalance.

Value

A ggplot2 object showing either:

- A boxplot for continuous variables (more than 4 unique values).
- A proportional bar chart for categorical variables (2–4 unique values).

Examples

```
dat   <- data.frame(
  cov1  = rnorm(50, 10, 1),
  cov2  = rnorm(50, 7,  1),
  cov3  = rnorm(50, 5,  1),
  treated = rep(c(0, 1), c(35, 15))
)
  out <- VCG_sampler(treated ~ cov1 + cov2 + cov3, data=dat, n=5, plot=FALSE)
  plot_var(out, what='cov1', group='VCG')
  plot_var(out, what='cov2', group='VCG')
```

---

robust_scale *Robust Scaling of Numeric and Categorical Variables*

Description

Applies robust scaling to numeric and categorical variables. For numeric variables, the function centers by the median and scales by the MAD. For categorical variables with 2–4 unique levels, it applies a custom transformation to map them to numeric values.

Usage

```
robust_scale(x, group = NULL)
```

Arguments

x	A numeric vector, factor, matrix, or data frame. If a matrix or data frame is provided, scaling is applied column-wise.
group	Optional vector indicating which group is the TG to scale to. If 'NULL' (default), centering and scaling are computed from all observations.

Details

This function is designed to make numeric and categorical variables comparable. This is an internal function that should not be used by package users.

Value

A scaled numeric vector or a data frame with scaled columns.

Examples

```
dat<-data.frame(x=rnorm(100, 10, 3), sex=factor(rbinom(100, 1, 0.5), labels=c("M","F")))

x<- robust_scale(dat$x, dat$sex)
round(median(x), 2)
round(mad(x), 2)
```

`VCG_sampler` *VCG Sampler for Energy Distance Balancing*

Description

This function performs energy distance based balancing and selects a subset from pool based on energy distance to approximate a randomized control trial. Optionally, it visualizes the balancing results.

Usage

```
VCG_sampler(formula, data, n, c_w = NULL, random = FALSE, plot = TRUE)
```

Arguments

`formula`	A formula specifying the treated indicator and covariates, e.g., 'treated ~ cov1 + cov2 | stratum'. The treated variable must be binary (0=pool, 1=treated)
`data`	A data frame containing the variables specified in the formula.
`n`	Integer. Number of observations to sample from the pool, or a vector of n for each stratum
`c_w`	Optional: Vector of positive weights for covariates, reflecting the relative importance of the covariates for balancing.
`random`	Logical. If 'TRUE', the distance is used as the probability for selecting the observation; otherwise, the nearest observations are used (deterministic). Default: FALSE
`plot`	Logical. If 'TRUE', returns a visualization of the balancing effect.

Details

If random is set to FALSE, the function selects the top ‘n‘ units from the pool with the lowest energy distance and assigns them to the VCG group. If random is set to TRUE, the function samples ‘n‘ units from pool with sampling probability inversely proportional to energy distance. The quality of covariate balancing is visualized using differences in medians and median absolute deviations (MADs). Permutation ellipses are generated by randomly permuting the pool and treated groups to estimate usual (random) variability. Only the X and Y axes are computed directly; the ellipse is interpolated between the axes. This method is intended as a visual approximation rather than a precise statistical test.

Value

If ‘plot = TRUE‘, returns a list with:

- A data frame with added columns:
  - ‘VCG‘: Indicator for selected pool units. VCG==1 indicates the VCG selected.
  - ‘e_weights‘: Energy weights used for selection
  - ‘<treated>_balanced‘: A factor indicating balanced treated assignment.
- A ggplot2 object showing the median and MAD differences before and after balancing, with a 95

If ‘plot = FALSE‘, returns only the modified data frame.

Examples

```
dat   <- data.frame(
  cov1  = rnorm(50, 10, 1),
  cov2  = rnorm(50, 7,  1),
  cov3  = rnorm(50, 5,  1),
  treated = rep(c(0, 1), c(35, 15))
)
  VCG_sampler(treated ~ cov1 + cov2 + cov3, data=dat, n=5)
```

Index

BestVCGsize, 2

combine_data, 3
combine_variables, 4

element, *7*
energy_distance, 5
energy_test, 6

find_outliers, 7

launchApp, 8

multiSampler, 9

plot_var, 10

robust_scale, 11
runApp, *9*

VCG_sampler, 12