Package: randomizationInference (via r-universe)

November 6, 2026

Version 1.0.4

Date 2022-05-17

Title Flexible Randomization-Based Inference

Author Joseph J. Lee and Tirthankar Dasgupta

Maintainer Joseph J. Lee <joseph.j.lee@post.harvard.edu>

Imports permute (>= 0.7-8), matrixStats, graphics, stats

Description Allows the user to conduct randomization-based inference for a wide variety of experimental scenarios. The package leverages a potential outcomes framework to output randomization-based p-values and null intervals for test statistics geared toward any estimands of interest, according to the specified null and alternative hypotheses. Users can define custom randomization schemes so that the randomization distributions are accurate for their experimental settings. The package also creates visualizations of randomization distributions and can test multiple test statistics simultaneously.

License GPL-2

NeedsCompilation no

Repository https://cran.r-universe.dev

Date/Publication 2022-05-17 19:00:02 UTC

RemoteUrl https://github.com/cran/randomizationInference

RemoteRef HEAD

RemoteSha 7abaf184368540e342625c79baca80a6e5952224

Contents

anovaF 2
blockRand 3
completeRand 4

constEffect 5
diffMeans 6
diffMeansVector 7
latinRand 8
randInterval 9
randomizationInference 11
randPlot 14
randTest 16
reading 21
withinBlockEffects 22
zeroEffect 23

Index **25**

---

anovaF *Analysis of Variance F statistic*

Description

Calculates the analysis of variance F statistic.

Usage

```
anovaF(y, w, calcOptions = NULL)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector or matrix of assignments.
calcOptions	a list of options for calculating the analysis of variance (ANOVA) F statistic (if necessary). `calcOptions$block` can denote a block variable to be accounted for. `calcOptions$row` and `calcOptions$col` can denote rows and columns to be accounted for in a Latin square design.

Details

Returns the F statistic calculated in an analysis of variance of a linear model with no interaction terms.

Value

An analysis of variance F statistic.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

Examples

```
# 1 treatment factor with 3 levels
# Assignments and outcomes
w <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 8, 12, 9, 9)
anovaF(y, w)

# 1 treatment factor with 3 levels, with block
# Assignments, blocks, and outcomes
w <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
x <- c(1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2)
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 8, 12, 9, 9)
anovaF(y, w, calcOptions = list(block = x))
```

---

blockRand *Random Treatment Assignments for Randomized Block Designs*

Description

Randomly draws a specified number of assignment vectors or matrices according to a randomized block design.

Usage

```
blockRand(w, nrand, block)
```

Arguments

w	a vector or matrix of assignments.
nrand	a number specifying the desired number of random assignments.
block	a vector of block designations.

Details

Assignments are randomly permuted within each block.

If w is a matrix, the permutations occur by row.

Value

A list of random assignment vectors or matrices.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

completeRand, latinRand

Examples

```
w1 <- c(0, 1, 0, 1, 0, 1, 0, 1)
block <- c(0, 0, 0, 0, 1, 1, 1, 1)
blockRand(w1, nrand = 5, block)

w2 <- c(0, 0, 1, 1, 0, 0, 1, 1)
blockRand(w = cbind(w1, w2), nrand = 5, block)
```

---

completeRand *Random Treatment Assignments for Completely Randomized Designs*

Description

Randomly draws a specified number of assignment vectors or matrices according to a completely randomized design.

Usage

```
completeRand(w, nrand)
```

Arguments

w	a vector or matrix of assignments.
nrand	a number specifying the desired number of random assignments.

Details

If w is a matrix, the permutations occur by row.

Value

A list of random assignment vectors or matrices.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

blockRand, latinRand

Examples

```
w1 <- c(0, 0, 0, 0, 1, 1, 1, 1)
completeRand(w1, nrand = 5)

w2 <- c(0, 1, 0, 1, 0, 1, 0, 1)
completeRand(w = cbind(w1, w2), nrand = 5)
```

constEffect *Potential Outcomes With Constant Treatment Effects*

Description

Calculates potential outcomes under modified assignments, according to the specified constant treatment effect(s).

Usage

```
constEffect(y, w, w_new, poOptions)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector or matrix of assignments.
w_new	a vector or matrix of modified assignments.
poOptions	a list of options for calculating potential outcomes. `poOptions$tau` is a number or numeric vector denoting the constant treatment effect(s).

Value

A vector of potential outcomes under the modified assignments.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

zeroEffect

Examples

```
# 1 treatment factor with 2 levels
# Assignments
w <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
# Modified Assignments
w_new <- c(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)
# Outcomes
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 8, 12)
constEffect(y, w, w_new, poOptions = list(tau = 2))

# 2 treatment factors, each with 2 levels
# Assignments
w1 <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
w2 <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
w <- cbind(w1, w2)
```

```
# Modified assignments
w1_new <- c(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)
w2_new <- c(1, 0, 1, 0, 1, 0, 1, 0, 1, 0)
w_new <- cbind(w1_new, w2_new)
# Outcomes
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 8, 12)
constEffect(y, w, w_new, poOptions = list(tau = c(2, -1)))
```

`diffMeans` *Single Pairwise Difference of Mean Outcomes*

Description

Calculates the difference of mean observed outcomes for a specified treatment factor and specified pair of comparison levels.

Usage

```
diffMeans(y, w, calcOptions = NULL)
```

Arguments

`y`	a vector or matrix of outcomes.
`w`	a vector or matrix of assignments.
`calcOptions`	a list of options for calculating the difference of mean outcomes (if necessary). `calcOptions$factor` is a number denoting the treatment factor of interest (defaults to 1). `calcOptions$pair` is a vector (of length 2) denoting the pair of levels for comparison (defaults to `c(0,1)`). If `calcOptions` is `NULL`, then `calcOptions$factor` and `calcOptions$pair` take on their default values.

Value

The difference of mean observed outcomes.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

`diffMeansVector`

Examples

```
# 1 treatment factor with 2 levels
# Assignments and outcomes
w <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
y <- c(4, 6, 5, 6, 4, 11, 11, 9, 10, 9)
diffMeans(y, w) # Equals 5

# 2 treatment factors, each with 3 levels
# Assignments and outcomes
w1 <- c(1, 2, 3, 1, 2, 3, 1, 2, 3)
w2 <- c(1, 2, 3, 2, 3, 1, 3, 1, 2)
w <- cbind(w1, w2)
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 10)
diffMeans(
  y,
  w,
  calcOptions = list(factor = 2, pair = c(1, 3))
) # Equals 0
```

---

diffMeansVector *Vector of Pairwise Differences of Mean Outcomes*

Description

Calculates the differences of mean observed outcomes for multiple specified treatment factors and specified pairs of comparison levels.

Usage

```
diffMeansVector(y, w, calcOptions)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector or matrix of assignments.
calcOptions	a list of options for calculating the vector of differences of mean outcomes. `calcOptions$factors` is a numeric vector denoting the treatment factors of interest. `calcOptions$pairs` is a matrix of pairs (specified by row) of levels for comparison.

Details

If unspecified, `calcOptions$pairs` defaults to `c(0,1)`.

If `calcOptions$factors` is specified, its length must equal the number of rows specified in `calcOptions$pairs`. If unspecified, `calcOptions$factors` defaults to an appropriately-sized vector of 1's.

Value

A vector of differences of mean observed outcomes.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

diffMeans

Examples

```
# 2 treatment factors, each with 3 levels
# Assignments, outcomes, and desired pairs
w1 <- c(1, 2, 3, 1, 2, 3, 1, 2, 3)
w2 <- c(1, 2, 3, 2, 3, 1, 3, 1, 2)
w <- cbind(w1, w2)
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 10)
diffMeansVector(
  y,
  w,
  calcOptions = list(
    factors = c(1, 1, 2),
    pairs = rbind(c(1, 2), c(2, 3), c(1, 3))
  )
) # Equals c(-1, 1, 0)
```

---

latinRand *Random Treatment Assignments for Isomorphic Latin Square Designs*

Description

Randomly draws an assignment vector or matrix according to the isomorphic Latin square design, for a specified number of permutations.

Usage

```
latinRand(w, nrand, row, col)
```

Arguments

w	a vector or matrix of assignments.
nrand	a number specifying the desired number of random assignments.
row	a vector of row designations.
col	a vector of column designations.

Details

Assignments are randomly permuted along rows and columns such that the Latin square design is preserved.

If w is a matrix, the permutations occur by row.

Value

A list of random isomorphic assignment vectors or matrices.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

completeRand, blockRand

Examples

```
w <- c(
  "C", "D", "B", "A", "A", "B", "D", "C",
  "D", "C", "A", "B", "B", "A", "C", "D"
)
row <- rep(1:4, 4)
col <- c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4))
latinRand(w, nrand = 5, row, col)
```

---

randInterval *Randomization-Based Null Interval*

Description

Calculates randomization-based intervals under the null hypothesis for the specified test statistics and coverage levels.

Usage

```
randInterval(results, coverage = 0.95)
```

Arguments

results	a resultant object of the randTest function.
coverage	a number specifying the desired coverage level (defaults to 0.95).

Details

If multiple tests are conducted simultaneously, users should be wary of multiple comparisons and make adjustments accordingly (e.g., Bonferroni corrections).

Value

A randomization-based interval (or multiple intervals) for the test statistic(s) under the null hypothesis with the specified coverage level.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

randTest, randPlot

Examples

```
# Completely randomized design example
# with one treatment factor at two levels
w <- c(rep(0, 5), rep(1, 5))
y <- rnorm(10, mean = 0, sd = 1)
# Two-sided test
twoSidedTest <- randTest(y, w, nrand = 50, calcTestStat = diffMeans)
randInterval(twoSidedTest)

# Reading comprehension pre- and post-test example
data(reading)
# Testing within-block pairwise effects
readingTest <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = withinBlockEffects,
  calcOptions = list(
    block = reading$Block,
    pairs = rbind(
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat"),
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat")
    ),
    blockindex = c(rep(1, 3), rep(2, 3))
  ),
  randOptions = list(type = "block", block = reading$Block)
)
randInterval(readingTest)
```

randomizationInference

Flexible Randomization-Based Inference

Description

Randomization-based p-values and null intervals for a wide variety of experimental scenarios, with corresponding visualizations.

Details

Package:	randomizationInference
Type:	Package
Version:	1.0.4
Date:	2022-05-17
License:	GPL-2
Main functions:	randTest, randPlot, randInterval

The randomizationInference package conducts randomization-based inference for a wide variety of experimental scenarios. The package leverages a potential outcomes framework to output randomization-based p-values and null intervals for test statistics geared toward any estimands of interest, according to the specified null and alternative hypotheses. Users can define custom randomization schemes so that the randomization distributions are accurate for their experimental settings. The package also creates visualizations of randomization distributions and can test multiple test statistics simultaneously.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

Maintainer: Joseph J. Lee <joseph.j.lee@post.harvard.edu>

References

Wu, C. F. J. and Hamada, M. (2009) Experiments, Planning, Analysis and Optimization (2nd ed), Wiley.

Moore, David S., and George P. McCabe (1989). Introduction to the Practice of Statistics.

Examples

```
# Completely randomized design example
# with one treatment factor at two levels
w <- c(rep(0, 5), rep(1, 5))
y <- rnorm(10, mean = 0, sd = 1)
# Two-sided test
twoSidedTest <- randTest(y, w, nrand = 50, calcTestStat = diffMeans)
```

```
randInterval(twoSidedTest)
randPlot(twoSidedTest)
# One-sided test
oneSidedTest <- randTest(
  y, w,
  nrand = 50,
  calcTestStat = diffMeans,
  alternative = "greater"
)
# Two=sided test with non-zero null hypothesis
nonZeroTest <- randTest(
  y,
  w,
  nrand = 50,
  calcTestStat = diffMeans,
  calcPO = constEffect,
  poOptions = list(tau = 2),
  null = 2
)

# Randomized block design example
# with one treatment factor at three levels
x <- rep(1:3, 4)
w_block <- rep(1:4, 3)
y_block <- rnorm(12, mean = x, sd = 1)
blockTest <- randTest(
  y_block,
  w_block,
  nrand = 50,
  calcTestStat = anovaF,
  calcOptions = list(block = x),
  randOptions = list(type = "block", block = x)
)
randInterval(blockTest)
randPlot(blockTest)

# 4x4 Latin square example (from the Wu/Hamada reference)
row <- rep(1:4, 4)
col <- c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4))
w_latin <- c(
  "C", "D", "B", "A", "A", "B", "D", "C",
  "D", "C", "A", "B", "B", "A", "C", "D"
)
y_latin <- c(
  235, 236, 218, 268, 251, 241, 227, 229,
  234, 273, 274, 226, 195, 270, 230, 225
)
latinTest <- randTest(
  y_latin,
  w_latin,
  nrand = 50,
  calcTestStat = anovaF,
  calcOptions = list(row = row, col = col),
```

```
  randOptions = list(type = "Latin", row = row, col = col)
)
randInterval(latinTest)
randPlot(latinTest)

# User-defined randomization example
# Partial randomization: first four assignments are fixed
# Due to physical limitations
# User-defined randomization function
# Input: number of random assignments, function options
# Output: list of random assignments
myRand <- function(nrand, userOptions = NULL){
  w_fixed = c(0, 0, 1, 1)
  lapply(1:nrand, function(i) c(w_fixed, sample(rep(0:1, 5))))
}
w_user <- c(c(0, 0, 1, 1), c(0, 1, 1, 0, 0, 0, 1, 1, 0, 1)) # observed assignment
y_user <- rnorm(14, mean = 0, sd = 1)
userTest <- randTest(
  y_user,
  w_user,
  nrand = 50,
  calcTestStat = diffMeans,
  randOptions = list(type = "user.defined"),
  userRand = myRand
)
randInterval(userTest)
randPlot(userTest)

# 2^3 factorial design example
# three treatment factors (OT, CP, and ST) at two levels each
OT <- c(-1, -1, -1, -1, 1, 1, 1, 1)
CP <- c(-1, -1, 1, 1, -1, -1, 1, 1)
ST <- rep(c(-1, 1), 4)
w_fac <- cbind(OT, CP, ST)
y_fac <- c(67, 79, 61, 75, 59, 90, 52, 87)
# Testing the main effect of factor "OT"
facTest1 <- randTest(
  y_fac,
  w_fac,
  nrand = 50,
  calcTestStat = diffMeans,
  calcOptions = list(factor = 1, pair = c(-1, 1))
)
# Testing all three main effects simultaneously
facTest2 <- randTest(
  y_fac,
  w_fac,
  nrand = 50,
  calcTestStat = diffMeansVector,
  calcOptions = list(
    factors = 1:3,
    pairs = matrix(rep(c(-1, 1), 3), ncol = 2, byrow = TRUE)
  )
```

```
)
# Testing all contrasts simultaneously
w_facNew <- cbind(OT, CP, ST, OT*CP, OT * ST, CP * ST, OT * CP * ST)
facTest3 <- randTest(
  y_fac,
  w_facNew,
  nrand = 50,
  calcTestStat = diffMeansVector,
  calcOptions = list(
    factors = 1:7,
    pairs = matrix(rep(c(-1, 1), 7), ncol = 2, byrow = TRUE)
  )
)
randInterval(facTest3)
randPlot(facTest3, plotDim = c(2, 4))

# Reading comprehension pre- and post-test example
data(reading)
# Ignoring blocks
readingTest1 <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = anovaF
)
# Testing within-block pairwise effects
readingTest2 <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = withinBlockEffects,
  calcOptions = list(
    block = reading$Block,
    pairs = rbind(
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat"),
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat")
    ),
    blockindex = c(rep(1, 3), rep(2, 3))
  ),
  randOptions = list(type = "block", block = reading$Block)
)
randInterval(readingTest2)
randPlot(readingTest2, plotDim = c(2, 3))
```

---

randPlot *Randomization-Based Plot*

Description

Plots observed test statistics against their randomization distributions and against the randomization-based intervals for their corresponding estimands.

Usage

```
randPlot(
  results,
  coverage = 0.95,
  breaks = 10,
  plotDim = c(length(results$obs_stat), 1)
)
```

Arguments

`results`	a resultant object of the `randTest` function.
`coverage`	a number specifying the desired coverage level (defaults to `0.95`).
`breaks`	a value specifying the histogram break pattern (defaults to 10).
`plotDim`	a numeric vector (of length 2) specifying the desired plot dimensions (if necessary, defaults to `c(length(results$obs_stat), 1)`).

Details

One plot will be displayed for each test statistic specified by `results`.

Observed test statistic values are demarcated by solid red lines. Comparison values under the null hypothesis are demarcated by solid black lines. Randomization-based interval bounds for the specified coverage level are demarcated by dotted blue lines.

Setting `plotDim = c(a, b)` is equivalent to specifying `par(mfrow=c(a, b))`. Plot dimensions are automatically reset to `c(1, 1)` afterward. If multiple test statistics are tested simultaneously, `plotDim` may need to be specified suitably for the plots to be displayed.

If multiple tests are conducted simultaneously, users should be wary of multiple comparisons and make adjustments accordingly (e.g., Bonferroni corrections).

Value

The plot(s) described in *details* will be displayed.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

`randTest`, `randInterval`

Examples

```
# Completely randomized design example
# with one treatment factor at two levels
w <- c(rep(0, 5), rep(1, 5))
y <- rnorm(10, mean = 0, sd = 1)
# Two-sided test
twoSidedTest <- randTest(y, w, nrand = 50, calcTestStat = diffMeans)
randPlot(twoSidedTest)

# Reading comprehension pre- and post-test example
data(reading)
# Testing within-block pairwise effects
readingTest <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = withinBlockEffects,
  calcOptions = list(
    block = reading$Block,
    pairs = rbind(
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat"),
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat")
    ),
    blockindex = c(rep(1, 3), rep(2, 3))
  ),
  randOptions = list(type = "block", block = reading$Block)
)
randPlot(readingTest, breaks = 20, plotDim = c(2, 3))
```

`randTest` *Randomization-Based Hypothesis Testing*

Description

Conducts randomization-based hypothesis tests according to the specified test statistic(s), assignment mechanism, and null and alternative hypotheses.

Usage

```
randTest(
  y,
  w,
  nrand,
  calcTestStat,
  calcOptions = NULL,
```

```
  calcPO = zeroEffect,
  poOptions = NULL,
  randOptions = list(
    type = c("complete", "block", "Latin", "user.defined"),
    block = NULL,
    row = NULL,
    col = NULL
  ),
  userRand = NULL,
  userOptions = NULL,
  null = NULL,
  alternative = c("two.sided", "greater", "less")
)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector (or matrix, when multiple treatment factors are present) of assignments.
nrand	a number specifying the desired number of random hypothetical assignments.
calcTestStat	a function to calculate the specified test statistic(s).
calcOptions	a list of options for calculating the test statistic(s) (if necessary).
calcPO	a function to calculate potential outcomes (defaults to zeroEffect).
poOptions	a list of options for calculating potential outcomes (if necessary).
randOptions	a list of options governing the random assignment mechanism.
userRand	a function to generate random assignments (if necessary).
userOptions	a list of options governing the user-defined random assignment mechanism (if necessary).
null	a number or numeric vector specifying the comparison value(s) under the null hypothesis.
alternative	a character string specifying the alternative hypothesis, must be one of the following: "two.sided" (default), "greater", or "less".

Details

The inputs to calcTestStat are y, w, and calcOptions (if necessary). The output of calcTestStat is a number or numeric vector representing the test statistic value(s). Several common test statistics (diffMeans, anovaF, diffMeansVector) are built-in for convenience. calcTestStat can also be a custom function, as long as the inputs and output are as described above. If multiple test statistics are tested simultaneously, users should be wary of multiple comparisons and make appropriate adjustments (e.g., Bonferroni corrections).

The inputs to calcPO are y, w, w_new (a vector or matrix of modified assignments), and poOptions (if necessary). The output of calcPO is a vector of potential outcomes under the modified assignments. Two common potential outcome calculations (zeroEffect (default) and constEffect) are built-in for convenience. calcPO can also be a custom function, as long as the inputs and output are as described above.

If unspecified, randOptions$type defaults to "complete". If randOptions$type equals "block", then randOptions$block must be specified. If randOptions$type equals "Latin", then randOptions$row and randOptions$col must be specified. If randOptions$type equals "user.defined", then userRand must be specified.

The inputs to userRand are nrand and userOptions (if necessary).

If null is specified, its length must equal the length of the output of calcTestStat. If unspecified, null defaults to 0 or an appropriately sized vector of 0's.

alternative = "greater" is the alternative that the test statistic is greater than the comparison value.

Value

A list containing the following elements:

perm_stats	a vector or matrix of hypothetical test statistic values.
obs_stat	the observed test statistic value(s).
null	a number or numeric vector specifying the comparison value(s) under the null hypothesis.
alternative	a character string specifying the alternative hypothesis.
pvalue	the randomization-based p-value(s).

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

References

Wu, C. F. J. and Hamada, M. (2009) Experiments, Planning, Analysis and Optimization (2nd ed), Wiley.

Moore, David S., and George P. McCabe (1989). Introduction to the Practice of Statistics. Original source: study conducted by Jim Baumann and Leah Jones of the Purdue University Education Department.

See Also

diffMeans, anovaF, diffMeansVector, zeroEffect, constEffect, completeRand, blockRand, latinRand, randInterval, randPlot

Examples

```
# Completely randomized design example
# with one treatment factor at two levels
w <- c(rep(0, 5), rep(1, 5))
y <- rnorm(10, mean = 0, sd = 1)
# Two-sided test
twoSidedTest <- randTest(y, w, nrand = 50, calcTestStat = diffMeans)
# One-sided test
oneSidedTest <- randTest(
```

```
  y,
  w,
  nrand = 50,
  calcTestStat = diffMeans,
  alternative = "greater"
)
# Two=sided test with non-zero null hypothesis
nonZeroTest <- randTest(
  y,
  w,
  nrand = 50,
  calcTestStat = diffMeans,
  calcPO = constEffect,
  poOptions = list(tau = 2),
  null = 2
)

# Randomized block design example
# with one treatment factor at three levels
x <- rep(1:3, 4)
w_block <- rep(1:4, 3)
y_block <- rnorm(12, mean = x, sd = 1)
blockTest <- randTest(
  y_block,
  w_block,
  nrand = 50,
  calcTestStat = anovaF,
  calcOptions = list(block = x),
  randOptions = list(type = "block", block = x)
)

# 4x4 Latin square example (from the Wu/Hamada reference)
row <- rep(1:4, 4)
col <- c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4))
w_latin <- c(
  "C", "D", "B", "A", "A", "B", "D", "C",
  "D", "C", "A", "B", "B", "A", "C", "D"
)
y_latin <- c(
  235, 236, 218, 268, 251, 241, 227, 229,
  234, 273, 274, 226, 195, 270, 230, 225
)
latinTest <- randTest(
  y_latin,
  w_latin,
  nrand = 50,
  calcTestStat = anovaF,
  calcOptions = list(row = row, col = col),
  randOptions = list(type = "Latin", row = row, col = col)
)

# 2^3 factorial design example
# three treatment factors (OT, CP, and ST) at two levels each
```

```
OT <- c(-1, -1, -1, -1, 1, 1, 1, 1)
CP <- c(-1, -1, 1, 1, -1, -1, 1, 1)
ST <- rep(c(-1, 1), 4)
w_fac <- cbind(OT, CP, ST)
y_fac <- c(67, 79, 61, 75, 59, 90, 52, 87)
# Testing the main effect of factor "OT"
facTest1 <- randTest(
  y_fac,
  w_fac,
  nrand = 50,
  calcTestStat = diffMeans,
  calcOptions = list(factor = 1, pair = c(-1, 1))
)
# Testing all three main effects simultaneously
facTest2 <- randTest(
  y_fac,
  w_fac,
  nrand = 50,
  calcTestStat = diffMeansVector,
  calcOptions = list(
    factors = 1:3,
    pairs = matrix(rep(c(-1, 1), 3), ncol = 2, byrow = TRUE)
  )
)
# Testing all contrasts simultaneously
w_facNew <- cbind(OT, CP, ST, OT * CP, OT * ST, CP * ST, OT * CP * ST)
facTest3 <- randTest(
  y_fac,
  w_facNew,
  nrand = 50,
  calcTestStat = diffMeansVector,
  calcOptions = list(
    factors = 1:7,
    pairs = matrix(rep(c(-1, 1), 7), ncol = 2, byrow = TRUE)
  )
)

# Reading comprehension pre- and post-test example
data(reading)
# Ignoring blocks
readingTest1 <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = anovaF
)
# Testing within-block pairwise effects
readingTest2 <- randTest(
  y = reading$Diff1,
  w = reading$Group,
  nrand = 50,
  calcTestStat = withinBlockEffects,
  calcOptions = list(
```

```
    block = reading$Block,
    pairs = rbind(
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat"),
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat")
    ),
    blockindex = c(rep(1, 3), rep(2, 3))
  ),
  randOptions = list(type = "block", block = reading$Block)
)
```

`reading` *Reading Data*

Description

Pre- and post-treatment reading comprehension test scores for 66 students randomly assigned to one of three methods for teaching reading comprehension.

Usage

```
data(reading)
```

Format

A data set of 66 students with 6 variables:

`Subject` student ID numbers

`Block` block designations

`Group` treatment group assignments

`Pre1` pre-treatment reading comprehension test scores

`Post1` post-treatment reading comprehension test scores

`Diff1` changes in test score (improvement if positive)

Source

Moore, David S., and George P. McCabe (1989). Introduction to the Practice of Statistics. Original source: study conducted by Jim Baumann and Leah Jones of the Purdue University Education Department.

Examples

```
data(reading)
```

withinBlockEffects *Pairwise Differences of Mean Outcomes Within Blocks*

Description

Calculates the differences of mean outcomes for multiple specified treatment factors and specified pairs of comparison levels, within the specified blocks.

Usage

```
withinBlockEffects(y, w, calcOptions)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector or matrix of assignments.
calcOptions	a list of options for calculating the difference of mean outcomes within blocks. calcOptions$block is a vector denoting the block designations. The numeric vector calcOptions$factors denotes the treatment factors of interest. calcOptions$pairs is a matrix of pairs (specified by row) of levels for comparison. calcOptions$blockindex is a vector of indices denoting the blocks within which the pairs should be compared.

Details

calcOptions$block should have the same length as y and w.

If unspecified, calcOptions$pairs defaults to c(0,1).

If calcOptions$factors is specified, its length must equal the number of rows specified in calcOptions$pairs. If unspecified, calcOptions$factors defaults to an appropriately-sized vector of 1's.

calcOptions$blockindex should have the same length as calcOptions$factors.

Value

A vector of differences of mean outcomes within blocks.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

References

Moore, David S., and George P. McCabe (1989). Introduction to the Practice of Statistics. Original source: study conducted by Jim Baumann and Leah Jones of the Purdue University Education Department.

See Also

diffMeansVector

Examples

```
# Reading comprehension pre- and post-test example
data(reading)
withinBlockEffects(
  y = reading$Diff1,
  w = reading$Group,
  calcOptions = list(
    block = reading$Block,
    pairs = rbind(
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat"),
      c("Basal", "DRTA"),
      c("Basal", "Strat"),
      c("DRTA", "Strat")
    ),
    blockindex = c(rep(1, 3), rep(2, 3))
  )
)
```

---

zeroEffect *Potential Outcomes With Zero Treatment Effects*

Description

Calculates potential outcomes under modified assignments, assuming zero treatment effect(s).

Usage

```
zeroEffect(y, w, w_new)
```

Arguments

y	a vector or matrix of outcomes.
w	a vector or matrix of assignments.
w_new	a vector or matrix of modified assignments.

Value

A vector of potential outcomes under the modified assignments.

Author(s)

Joseph J. Lee and Tirthankar Dasgupta

See Also

constEffect

Examples

```
# Assignments
w <- c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)
# Modified Assignments
w_new <- c(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)
# Outcomes
y <- c(4, 6, 5, 7, 4, 7, 11, 9, 8, 12)
zeroEffect(y, w, w_new) # Returns y as is
```

Index

∗ **datasets**
reading, 21
∗ **methods**
anovaF, 2
blockRand, 3
completeRand, 4
constEffect, 5
diffMeans, 6
diffMeansVector, 7
latinRand, 8
randInterval, 9
randPlot, 14
randTest, 16
withinBlockEffects, 22
zeroEffect, 23
∗ **package**
randomizationInference, 11

anovaF, 2, *17, 18*

blockRand, 3, *4*, *9*, *18*

completeRand, *3*, 4, *9*, *18*
constEffect, 5, *17, 18*, *24*

diffMeans, 6, *8*, *17, 18*
diffMeansVector, *6*, 7, *17, 18*, *23*

latinRand, *3, 4*, 8, *18*

randInterval, 9, *11*, *15*, *18*
randomizationInference, 11
randomizationInference-package
*(*randomizationInference*)*, 11
randPlot, *10, 11*, 14, *18*
randTest, *10, 11*, *15*, 16
reading, 21

withinBlockEffects, 22

zeroEffect, *5*, *17, 18*, 23