Package 'IsoCheck'

January 20, 2025

Title Isomorphism Check for Multi-Stage Factorial Designs with Randomization Restrictions

Version 0.1.0

Date 2020-03-20

Author Neil Spencer [aut, cre], Pritam Ranjan [aut,cre], Franklin Mendivil [ctb]

Maintainer Pritam Ranjan <pritam.ranjan@gmail.com>

Description Contains functions to check the isomorphism of multi-stage factorial designs with randomisation restrictions based on balanced spreads and balanced covering stars of PG(n-1,2) as described in Spencer, Ranjan and Mendivil (2019) <doi:10.1007/s42519-019-0064-5>.

License GPL-2

Imports gtools, dplyr, plyr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-25 15:40:02 UTC

Contents

`applyCollineation` *Relabel a Spread or Star with a Collineation Matrix*

Description

This function relabels a balanced (t-1)-spread or a covering star $St(n, \mu, t, t_0)$ of `PG(n-1,2)` according to the specified collineation matrix.

Usage

```
applyCollineation(C, spr)
```

Arguments

`C`	A binary n by n matrix representing a collineation of `PG(n-1,2)`.
`spr`	A balanced spread or star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).

Details

This code applies the relabelling corresponding to a collineation matrix C to any given balanced spread or star of PG(n-1, 2). The spread should be formatted as a 3-dimensional array with `spr[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `spr`. The collineation is applied via a matrix multiplication modulo 2 (i.e., the calculations are done over GF(2)). See Spencer et al. (2019) for details.

Value

A spread or star of the same dimensions as spr.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkSpreadIsomorphism for checking the isomorphism of balanced spreads.
checkStarIsomorphism for checking the isomorphism of balanced covering stars.

Examples

```
## Example 1: relabelling a 1-spread of PG(3,2)
data(spreadn4t2a)
Collin <- cbind(c(1,0,0,1), c(0,0,1,1), c(1,1,1,1), c(0,1,1,1))
# Collin is the collineation matrix corresponding to
# A -> AD, B -> CD, C -> ABCD, D -> BCD
applyCollineation(Collin, spreadn4t2a)



## Example 2: Relabelling a star of PG(4,2) consisting of 4-flats.
data(starn5t3a)
Collin2 <- cbind(c(0,0,0,0,1), c(1,0,0,0,0), c(0,1,0,0,0), c(0,0,0,1,0), c(0,0,1,0,0))
# Collin2 is the collineation matrix corresponding to
# A -> E, B -> A, C -> B, D -> D, E -> C
applyCollineation(Collin2, starn5t3a)
```

---

checkSpreadEquivalence

Checking the Equivalence of Two Spreads

Description

This function checks the equivalence of two (t-1)-spreads of PG(n-1,2) by comparing their sorted bitstring representations.

Usage

```
checkSpreadEquivalence(spread1, spread2)
```

Arguments

spread1	A (t-1)-spread of PG(n-1,2) stored as a three dimensional binary array (see Details and Examples).
spread2	A (t-1)-spread of PG(n-1,2) stored as a three dimensional binary array (see Details and Examples).

Details

This code checks if two (t-1)-spreads of PG(n-1,2) are equivalent using the bitstring representation of Spencer et al. (2019). Both input spreads should be formatted as 3-dimensional arrays, for example, spread1[i,j,k] indicates whether or not the ith basic factor is present in the jth effect of the kth flat of spread1.

Value

A Boolean indicating whether or not the two spreads are equivalent.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkSpreadIsomorphism for checking the isomorphism of spreads.
checkStarEquivalence for checking the equivalence of two stars.

Examples

```
## Example 1: two non-equivalent 1-spreads of PG(3,2)
data(spreadn4t2a)
data(spreadn4t2b)

# test their equivalence
(test1 <- checkSpreadEquivalence(spreadn4t2a, spreadn4t2b))


# direct instantiation of a spread
spreadn4t2c <- array(NA, c(4,3,5))

spreadn4t2c[,1,1] <- c(0, 0, 0, 1)
spreadn4t2c[,2,1] <- c(0, 1, 1, 0)
spreadn4t2c[,3,1] <- c(0, 1, 1, 1)
spreadn4t2c[,1,2] <- c(0, 0, 1, 0)
spreadn4t2c[,2,2] <- c(1, 1, 0, 0)
spreadn4t2c[,3,2] <- c(1, 1, 1, 0)
spreadn4t2c[,1,3] <- c(0, 1, 0, 0)
spreadn4t2c[,2,3] <- c(1, 0, 1, 1)
spreadn4t2c[,3,3] <- c(1, 1, 1, 1)
spreadn4t2c[,1,4] <- c(1, 0, 0, 0)
spreadn4t2c[,2,4] <- c(0, 1, 0, 1)
spreadn4t2c[,3,4] <- c(1, 1, 0, 1)
spreadn4t2c[,1,5] <- c(0, 0, 1, 1)
spreadn4t2c[,2,5] <- c(1, 0, 1, 0)
spreadn4t2c[,3,5] <- c(1, 0, 0, 1)

(test2 <- checkSpreadEquivalence(spreadn4t2a, spreadn4t2c))
```

```
## Example 2: two equivalent 2-spreads of PG(5,2)
data(spreadn6t3a)

# permute the flats and flat order of spreadn6t3a to create a
# second equivalent spread equiv_spreadn6t3a.

equiv_spreadn6t3a <- spreadn6t3a
dims <- dim(equiv_spreadn6t3a)
for(i in 1:(dims[3])){
   equiv_spreadn6t3a[,,i] <- equiv_spreadn6t3a[,sample(1:dims[2], dims[2]),i]
}
equiv_spreadn6t3a <- equiv_spreadn6t3a[,,sample(1:dims[3], dims[3])]

(test3 <- checkSpreadEquivalence(spreadn6t3a, equiv_spreadn6t3a))
```

`checkSpreadIsomorphism`

Checking the Isomorphism of Two Spreads

Description

This function checks the isomorphism of two `(t-1)`-spreads of `PG(n-1,2)`. If they are isomorphic, it returns the list of isomorphism establishing collineations (IECs). The option is provided to enumerate all IECs or to terminate after the first one is found.

Usage

```
checkSpreadIsomorphism(spread1, spread2, returnfirstIEC = FALSE, printstatement = TRUE)
```

Arguments

`spread1`	A `(t-1)`-spread of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).
`spread2`	A `(t-1)`-spread of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).
`returnfirstIEC`	An indicator to indicate whether all isomorphism establishing collineations should be returned (default), or terminate only after the first one is found.
`printstatement`	If set to true (default), running the function also prints a sentence declaring the isomorphism of the spreads.

Details

This code considers all possible collineations of `PG(n-1,2)` to search for isomorphism establishing collineations (IECs) from spread1 to spread2. The search is conducted over the reduced space described in Algorithm 1 of Spencer et al. (2019). Equivalence is assessed using the bitstring comparison method described in Spencer et al. (2019).

Both input spreads should be formatted as 3-dimensional arrays with `spread1[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of spread1.

Value

A list containing two objects. The first object is a Boolean indicating whether or not spread1 is isomorphic to spread2. If isomorphic, the second object is a list of isomorphism establishing collineation matrices. If not isomorphic, the second object is NA.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.
`checkSpreadEquivalence` for checking the equivalence of balanced spreads.

Examples

```
## Example 1: two 1-spreads of PG(3,2)
data(spreadn4t2a)
data(spreadn4t2b)
# test their isomorphism
test1 <- checkSpreadIsomorphism(spreadn4t2a, spreadn4t2b)
test1$result # the test indicates that they are isomorphic
(IEC1 <- (test1$IECs)[[1]])
# we store the first isomorphism establishing collineation as IEC1



## Example 2: two 2-spreads of PG(5,2) using returnfirstIEC to cut down on runtime
data(spreadn6t3a)
data(spreadn6t3b)
test2 <- checkSpreadIsomorphism(spreadn6t3a, spreadn6t3b, returnfirstIEC = TRUE)
test2$result # the test indicates that they are isomorphic



## Example 3: non-isomorphic 1-spreads of PG(5,2)
data(spreadn6t2a)
data(spreadn6t2c)

# A bit slow for official example
```

```
# test3 <- checkSpreadIsomorphism(spreadn6t2a, spreadn6t2c, returnfirstIE#C = TRUE)
# test3$result



## Example 4: isomorphic 1-spreads of PG(5,2)
data(spreadn6t2a)
data(spreadn6t2b)
test4 <- checkSpreadIsomorphism(spreadn6t2a, spreadn6t2b, returnfirstIEC = TRUE)
test4$result # the test indicates that they are isomorphic
```

`checkStarEquivalence` *Checking the Equivalence of Two Stars*

Description

This function checks the equivalence of two balanced covering stars of `PG(n-1,2)` by comparing sorted bitstring representations.

Usage

```
checkStarEquivalence(star1, star2)
```

Arguments

`star1`	A star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).
`star2`	A star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).

Details

This code checks if two stars of `PG(n-1,2)` are equivalent using the bitstring representation of Spencer et al. (2019). Both input stars should be formatted as 3-dimensional arrays with `spread1[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of spread1.

Value

A Boolean indicating whether or not the two stars are equivalent.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkStarIsomorphism for checking the isomorphism of stars.
checkSpreadEquivalence for checking the equivalence of spreads.

Examples

```
## Example 1: two non-equivalent stars of PG(4,2)
data(starn5t3a)
data(starn5t3b)

# test their equivalence
(test1 <- checkStarEquivalence(starn5t3a, starn5t3b))




## Example 2: two equivalent stars of PG(7,2) consisting of 6-flats
data(starn8t5a)

#permute the flats and flat order of starn8t5a to create a second equivalent spread equiv_starn8t5a.
equiv_starn8t5a <- starn8t5a
dims <- dim(equiv_starn8t5a)
for(i in 1:(dims[3])){
   equiv_starn8t5a[,,i] <- equiv_starn8t5a[,sample(1:dims[2], dims[2]),i]
}
equiv_starn8t5a <- starn8t5a[,,sample(1:dims[3], dims[3])]

(test2 <- checkStarEquivalence(starn8t5a, equiv_starn8t5a))
```

---

checkStarIsomorphism *Checking the Isomorphism of Two Stars*

Description

This function checks the isomorphism of two balanced covering stars of PG(n-1,2). If they are isomorphic, it returns the list of isomorphism establishing collineations. The option is provided to enumerate all isomorphism establishing collineations or to terminate after the first one is found.

Usage

```
checkStarIsomorphism(star1, star2, returnfirstIEC = FALSE)
```

Arguments

star1	A star of PG(n-1,2) stored as a three dimensional binary array (see Details and Examples of checkSpreadEquivalence).
star2	A star of PG(n-1,2) stored as a three dimensional binary array (see Details and Examples of checkSpreadEquivalence).

`returnfirstIEC` An indicator of whether all isomorphism establishing collineations (IECs) should be returned (default), or terminate only after the first one is found.

Details

This code considers all possible collineations of `PG(n-1,2)` to search for isomorphism establishing collineations from star1 to star2. The search is conducted by first projecting onto a lower dimensional space described as Algorithm 2 in Spencer et al. (2019). Equivalence is assessed using the bitstring comparison method described in Spencer et al. (2019). Both input stars should be formatted as 3-dimensional arrays with `star1[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of star1.

Value

A list containing two objects. The first object is a Boolean indicating whether or not star1 is isomorphic to star2. If isomorphic, the second object is a list of isomorphism establishing collineation matrices. If not isomorphic, the second object is NA.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of spreads.
`checkStarEquivalence` for checking the equivalence of stars.

Examples

```
## Example 1: Two stars of PG(4,2) consisting of 4-flats
data(starn5t3a)
data(starn5t3b)
test1 <- checkStarIsomorphism(starn5t3a, starn5t3b, returnfirstIEC = TRUE)
test1$result # the test indicates that they are isomorphic
(IECstar <- test1$IECs[[1]]) # the first IEC




## Example 2: Two stars of PG(7,2) consisting of 6-flats
data(starn8t5a)
data(starn8t5b)
test2 <- checkStarIsomorphism(starn8t5a, starn8t5b, returnfirstIEC = TRUE)
test2$result
```

`getBitstrings` *Bitstring Representation of a Spread or Star*

Description

This function computes a bitstring representation for a spread or star of `PG(n-1,2)`

Usage

```
getBitstrings(spr)
```

Arguments

`spr`	A spread or star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).

Details

This code obtains the bitstring representation (as described in Spencer et al. 2019) for any given spread or star of `PG(n-1,2)`. The spread should be formatted as a 3-dimensional array with `spr[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `spr`. This representation facilitates fast equivalence checking for spreads or stars.

Value

A matrix with each row characterizing the elements of a distinct flat in spr.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadEquivalence` for checking equivalence of spreads.
`checkStarEquivalence` for checking equivalence of spreads.

Examples

```
## Example 1: The bitstring representation of a 1-spread of PG(3,2)
data(spreadn4t2a)
getBitstrings(spreadn4t2a)



## Example 2: The bitstring representation of a star of PG(4,2) consisting of 4-flats.
data(starn5t3a)
getBitstrings(starn5t3a)
```

`is.spread` *Boolean check for a proper Spread*

Description

Checks whether or not the input is a proper balanced (t-1)-spread of `PG(n-1,2)`.

Usage

```
is.spread(spr, printstatements = TRUE)
```

Arguments

`spr`	A balanced spread of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).
`printstatements`	A Boolean indicating whether or not to print possible reasons for not being a spread if the input is not a spread.

Details

Checks whether or not the input is a proper balanced (t-1)-spread of `PG(n-1,2)`. The spread should be formatted as a 3-dimensional array with `spr[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `spr`. See Spencer et al. (2019) for details.

Value

A Boolean indicating whether or not the input is a spread.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`is.star` for checking if the input is a balanced covering star.
`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

Examples

```
## Example 1: checking whether "spreadn4t2a" is a proper spread
data(spreadn4t2a)
is.spread(spreadn4t2a)



## Example 2: checking whether "starn5t3a" is a proper spread
data(starn5t3a)
is.spread(starn5t3a)
```

---

is.star *Boolean check for a proper Star*

Description

Checks whether or not the input is a proper balanced covering star of `PG(n-1,2)`.

Usage

```
is.star(star, printstatements = TRUE)
```

Arguments

`star`	A balanced covering star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).
`printstatements`	A Boolean indicating whether or not to print possible reasons for not being a star if the input is not a balanced covering star.

Details

Checks whether or not the input is a proper balanced covering star of `PG(n-1,2)`. The star should be formatted as a 3-dimensional array with `star[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `star`. See Spencer et al. (2019) for details.

Value

A Boolean indicating whether or not the input is a star.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`is.spread` for checking if the input is a balanced spread.
`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

Examples

```
## Example 1: checking whether "spreadn4t2a" is a proper star
data(spreadn4t2a)
is.star(spreadn4t2a)



## Example 2: checking whether "starn5t3a" is a proper star
data(starn5t3a)
is.star(starn5t3a)
```

---

spreadn4t2a *Data: A cyclic 1-spread of PG(3,2)*

Description

A balanced 1-spread of PG(3,2) obtained via cyclic construction

Usage

```
data(spreadn4t2a)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of five subspaces f_1, f_2, f_3, f_4, f_5 given by $f_1 = (D, BC, BCD)$, $f_2 = (C, AB, ABC)$, $f_3 = (B, ACD, ABCD)$, $f_4 = (A, BD, ABD)$ and $f_5 = (CD, AC, AD)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

---

spreadn4t2b *Data: A cyclic 1-spread of PG(3,2)*

Description

A balanced 1-spread of PG(3,2) obtained via cyclic construction

Usage

```
data(spreadn4t2b)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of five subspaces f_1, f_2, f_3, f_4, f_5 given by $f_1 = (A, CD, ACD)$, $f_2 = (C, ABCD, ABD)$, $f_3 = (D, B, BD)$, $f_4 = (ABC, AD, BCD)$ and $f_5 = (AC, AB, BC)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkSpreadIsomorphism for checking the isomorphism of balanced spreads.
checkStarIsomorphism for checking the isomorphism of balanced covering stars.

---

spreadn6t2a *Data: A cyclic 1-spread of PG(5,2)*

Description

A balanced 1-spread of PG(5,2) obtained via cyclic construction

Usage

```
data(spreadn6t2a)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of 21 subspaces $f_1, f_2, ..., f_{21}$ given by $f_1 = (F, ABCEF, ABCE)$, $f_2 = (E, ABDF, ABDEF)$, $f_3 = (D, ACF, ACDF)$, $f_4 = (C, BF, BCF)$, $f_5 = (B, AE, ABE)$, $f_6 = (A, DEF, ADEF)$, $f_7 = (EF, CDE, CDF)$, $f_8 = (DE, BCD, BCE)$, $f_9 = (CD, ABC, ABD)$, $f_{10} = (BC, ABEF, ACEF)$, $f_{11} = (AB, ADF, BDF)$, $f_{12} = (DF, BE, BDEF)$, $f_{13} = (CE, AD, ACDE)$, $f_{14} = (AC, BDE, ABCDE)$, $f_{15} = (BEF, ACD, ABCDEF)$, $f_{16} = (ADE, BCEF, ABCDF)$, $f_{17} = (CDEF, ABDE, ABCF)$, $f_{18} = (BCDE, ACDEF, ABF)$, $f_{19} = (ABCD, BCDF, AF)$, $f_{20} = (AEF, CF, ACE)$, and $f_{21} = (BD, CEF, BCDEF)$.

Note

In R, the data must be loaded using the data function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkSpreadIsomorphism for checking the isomorphism of balanced spreads.
checkStarIsomorphism for checking the isomorphism of balanced covering stars.

spreadn6t2b *Data: A cyclic 1-spread of PG(5,2)*

Description

A balanced 1-spread of PG(5,2) obtained via cyclic construction

Usage

```
data(spreadn6t2b)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of 21 subspaces $f_1, f_2, ..., f_{21}$ given by
$f_1 = (EF, BCD, BCDEF)$, $f_2 = (A, ACF, CF)$, $f_3 = (B, ABCDE, ACDE)$, $f_4 = (DF, F, D)$,
$f_5 = (E, BC, BCE)$, $f_6 = (ABC, ABEF, CEF)$, $f_7 = (AEF, ABDF, BDE)$, $f_8 = (AB, BDEF, ADEF)$,
$f_9 = (BDF, ABCDEF, ACE)$, $f_{10} = (DEF, BCF, BCDE)$, $f_{11} = (ABCE, ACEF, BF)$,
$f_{12} = (BEF, AE, ABF)$, $f_{13} = (ADF, AC, CDF)$, $f_{14} = (ABCDF, ABE, CDEF)$, $f_{15} =$
$(AF, ACDF, CD)$, $f_{16} = (C, AD, ACD)$, $f_{17} = (ABDE, CE, ABCD)$, $f_{18} = (ABDEF, CDE, ABCF)$,
$f_{19} = (ACDEF, BD, ABCEF)$, $f_{20} = (BCEF, DE, BCDF)$, and $f_{21} = (BE, ADE, ABD)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

spreadn6t2c *Data: A cyclic 1-spread of PG(5,2)*

Description

A balanced 1-spread of PG(5,2) obtained via cyclic construction

Usage

```
data(spreadn6t2c)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of 21 subspaces $f_1, f_2, ..., f_{21}$ given by $f_1 = (F, ABCEF, ABCE)$, $f_2 = (E, ABDF, ABDEF)$, $f_3 = (D, ACF, ACDF)$, $f_4 = (C, BF, BCF)$, $f_5 = (B, AE, ABE)$, $f_6 = (A, DEF, ADEF)$, $f_7 = (EF, CDE, CDF)$, $f_8 = (DE, BCD, BCE)$, $f_9 = (CD, ABC, ABD)$, $f_{10} = (BC, ABEF, ACEF)$, $f_{11} = (AB, ADF, BDF)$, $f_{12} = (DF, BE, BDEF)$, $f_{13} = (CE, AD, ACDE)$, $f_{14} = (AC, BDE, ABCDE)$, $f_{15} = (BEF, ACD, ABCDEF)$, $f_{16} = (ADE, BCEF, ABCDF)$, $f_{17} = (CDEF, ABDE, ABCF)$, $f_{18} = (BCDE, ACDEF, ABF)$, $f_{19} = (ACE, AF, CEF)$, $f_{20} = (CF, BD, BCDF)$, and $f_{21} = (ABCD, AEF, BCDEF)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

`spreadn6t3a` *Data: A cyclic 2-spread of PG(5,2)*

Description

A balanced 2-spread of PG(5,2) obtained via cyclic construction

Usage

```
data(spreadn6t3a)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of 9 subspaces $f_1, f_2, ..., f_9$ each of size 7. The subspaces are
$f_1 = (A, EF, AEF, BCE, ABCE, BCF, ABCF)$, $f_2 = (B, AF, ABF, CDF, BCDF, ACD, ABCD)$,
$f_3 = (C, AB, ABC, ADE, ACDE, BDE, BCDE)$, $f_4 = (D, BC, BCD, BEF, BDEF, CEF, CDEF)$,
$f_5 = (E, CD, CDE, ACF, ACEF, ADF, ADEF)$, $f_6 = (F, DE, DEF, ABD, ABDF, ABE, ABEF)$,
$f_7 = (BD, BF, DF, ACE, ABCDE, ABCEF, ACDEF)$,
$f_8 = (AC, CE, AE, BDF, ABCDF, BCDEF, ABDEF)$
and $f_9 = (AD, BE, ABDE, CF, ACDF, BCEF, ABCDEF)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

spreadn6t3b *Data: A cyclic 2-spread of PG(5,2)*

Description

A balanced 2-spread of PG(5,2) obtained via cyclic construction

Usage

```
data(spreadn6t3b)
```

Format

The spread is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the spread.

Details

The spread consists of 9 subspaces $f_1, f_2, ..., f_9$ each of size 7. The subspaces are
$f_1 = (ABC, AEF, BCEF, ADEF, BCDEF, D, ABCD)$, $f_2 = (E, ABCEF, ABCF, BDE, BD, ACDF, ACDEF)$,
$f_3 = (DF, ABCE, ABCDEF, C, CDF, ABE, ABDEF)$, $f_4 = (B, DEF, BDEF, AF, ABF, ADE, ABDE)$,
$f_5 = (A, BDF, ABDF, ABCDE, BCDE, ACEF, CEF)$, $f_6 = (EF, AB, ABEF, ACE, ACF, BCE, BCF)$,
$f_7 = (BE, F, BEF, BCDF, CDEF, BCD, CDE)$,
$f_8 = (ABCDF, ADF, BC, BF, ACD, ABD, CF)$ and
$f_9 = (AC, AE, CE, DE, ACDE, AD, CD)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

`starn5t3a` *Data: A 2-star of PG(4,2)*

Description

A balanced 2-star of PG(4,2)

Usage

```
data(starn5t3a)
```

Format

The star is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the star.

Details

The star consists of five subspaces f_1, f_2, f_3, f_4, f_5 given by $f_1 = (A, E, CDE, AE, ACD, ACDE, CD)$, $f_2 = (D, BC, BCD, AD, ABC, ABCD, A)$, $f_3 = (C, BDE, BCDE, AC, ABDE, ABCDE, A)$, $f_4 = (B, BCE, CE, AB, ACE, A, ABCE)$ and $f_5 = (DE, BD, BE, A, ABD, ABE, ADE)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

starn5t3b *Data: A 2-star of PG(4,2)*

Description

A balanced 2-star of PG(4,2)

Usage

```
data(starn5t3b)
```

Format

The star is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the star.

Details

The star consists of five subspaces f_1, f_2, f_3, f_4, f_5 given by $f_1 = (ABC, AC, CDE, B, BCDE, ABDE, ADE)$, $f_2 = (AE, DE, AD, BCE, ABCDE, BCD, ABC)$, $f_3 = (D, C, CD, ABCD, AB, ABD, ABC)$, $f_4 = (E, ACDE, ACD, ABCE, BD, ABC, BDE)$ and $f_5 = (CE, A, ACE, ABC, BC, BE, ABE)$.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

`starn8t5a` *Data: A 4-star of PG(7,2)*

Description

A balanced 4-star of PG(7,2)

Usage

```
data(starn8t5a)
```

Format

The star is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the star.

Details

The star consists of nine subspaces f_1, f_2, , ..., f_9 of size 31 each. Use `vectortostring(starn8t5a)` to see the elements of this star.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

starn8t5b *Data: A 4-star of PG(7,2)*

Description

A balanced 4-star of PG(7,2)

Usage

```
data(starn8t5b)
```

Format

The star is formatted as a 3-dimensional array, where the `[i,j,k]`-th element indicates whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of the star.

Details

The star consists of nine subspaces f_1, f_2, , ..., f_9 of size 31 each. Use `vectortostring(starn8t5b)` to see the elements of this star.

Note

In R, the data must be loaded using the `data` function.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

`star_to_spread` *Converts a star to its corresponding spread*

Description

A function that converts a balanced covering star `St(n, mu, t, t_0)` of `PG(n-1,2)` to its corresponding balanced `((t-t_0)-1)`-spread of `PG((n-t_0)-1,2)`.

Usage

```
star_to_spread(star)
```

Arguments

`star`	A balanced covering star of `PG(n-1,2)` stored as a three dimensional binary array (see Details and Examples of `checkSpreadEquivalence`).

Details

Finds a balanced `((t-t_0)-1)`-spread of `PG((n-t_0)-1,2)` embedded in `PG(n-1,2)` as conformable with the geometry of a balanced covering star `St(n, mu, t, t_0)` of `PG(n-1,2)`. The star should be formatted as a 3-dimensional array with `star[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `star`. See Spencer et al. (2019) for details.

Value

A balanced `((t-t_0)-1)`-spread of `PG((n-t_0)-1,2)`

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`is.spread` for checking if the input is a balanced spread.
`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

Examples

```
## Example 1: checking whether "starn8t5a" is a proper star
data(starn8t5a)
star_to_spread(starn8t5a)



## Example 2: checking whether "starn5t3a" is a proper star
data(starn5t3a)
star_to_spread(starn5t3a)
```

`stringtovector` *Converts a character string representation of a factorial effect into the vector form*

Description

Converts a character string representation of a factorial effect in `PG(n-1,2)` into a binary vector of length n.

Usage

```
stringtovector(string,n)
```

Arguments

`string`	a character string representation of a factorial effect in `PG(n-1,2)` (see Details and Examples of `checkSpreadEquivalence`).
`n`	the number of basic factors, or equivalently, the dimension of `PG(n-1,2)`

Details

Takes a character string representation of a factorial effect in `PG(n-1,2)`, and returns a binary vector of length n. This can be used in defining a spread or a star. The spread/star should be formatted as a 3-dimensional array with `spread[i,j,k]`/`star[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `spread`/`star`. See Spencer et al. (2019) for details.

Value

A binary vector of length n.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

`checkSpreadIsomorphism` for checking the isomorphism of balanced spreads.
`checkStarIsomorphism` for checking the isomorphism of balanced covering stars.

Examples

```
## Example : Converts "AC" into a vector representation
stringtovector("AC",4)
stringtovector("AC",5)
stringtovector("CD",6)
```

`vectortostring` *Converts a binary vector to character strings*

Description

Converts a binary vector or matrix in `PG(n-1,2)` into string-vector/matrix .

Usage

```
vectortostring(arry)
```

Arguments

`arry`	A binary vector or a matrix of binary vectors representing a flat or spread/star in `PG(n-1,2)` (see Details and Examples of `checkSpreadEquivalence`).

Details

Takes an array (a binary vector or an array of binary vectors, upto three dimensions) and returns the character string representation of the vectors. This can be used for reporting spreads and stars in an easy-to-read format. Recall that a sprad/star should be formatted as a 3-dimensional array with `star[i,j,k]` indicating whether or not the `i`th basic factor is present in the `j`th effect of the `k`th flat of `star`. See Spencer et al. (2019) for details.

Value

Character string representation of the input `arry`.

Author(s)

Neil Spencer, Pritam Ranjan, Franklin Mendivil

References

Spencer, N.A., Ranjan, P., and Mendivil, F., (2019), "Isomorphism Check for 2^n Factorial Designs with Randomization Restrictions", Journal of Statistical Theory and Practice, 13(60),1-24 [https://doi.org/10.1007/s42519-019-0064-5]

See Also

checkSpreadIsomorphism for checking the isomorphism of balanced spreads.
checkStarIsomorphism for checking the isomorphism of balanced covering stars.

Examples

```
## Example 1: converts c(0,1,1,0) into "BC"
vec = c(0,1,1,0)
vectortostring(vec)



## Example 2: converts "spreadn6t3a" into character string representation
data(spreadn6t3a)
vectortostring(spreadn6t3a)



## Example 3: converts "starn5t3a" into character string representation
data(starn5t3a)
vectortostring(starn5t3a)
```

Index