Package ‘yhat’

October 10, 2023

Type Package
Title Interpreting Regression Effects
Version 2.0-4
Date 2023-10-10
Author Kim Nimon <kim.nimon@gmail.com>, Fred Oswald, and J. Kyle Roberts.
Maintainer Kim Nimon <kim.nimon@gmail.com>
Depends R (>= 2.7.0)
Imports yacca, miscTools, plotrix, boot
Suggests MBESS
Description The purpose of this package is to provide methods to interpret multiple linear regression and canonical correlation results including beta weights, structure coefficients, validity coefficients, product measures, relative weights, all-possible-subsets regression, dominance analysis, commonality analysis, and adjusted effect sizes.
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2023-10-10 18:20:02 UTC

R topics documented:

yhat-package .................................................. 2
aps ............................................................... 3
boot.yhat ..................................................... 4
booteval.yhat ............................................... 5
calc.yhat .................................................... 7
canonCommonality ......................................... 8
canonVariate ............................................... 10
ci.yhat ....................................................... 11
combCI ....................................................... 12
commonality ............................................... 13
commonalityCoefficients ................................. 14
**Description**

The purpose of this package is to provide methods to interpret multiple linear regression and canonical correlation results including beta weights, structure coefficients, validity coefficients, product measures, relative weights, all-possible-subsets regression, dominance analysis, commonality analysis, and adjusted effect sizes.

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>, Fred L. Oswald, J. Kyle Roberts

**References**


**See Also**

regr commonalityCoefficients canonCommonality calc.yhat boot.yhat booteval.yhat plotCI.yhat aps commonality dominance dombin rlw
The function runs all possible subsets regression and returns data needed to run commonality and dominance analysis.

Usage

```r
aps(dataMatrix, dv, ivlist)
```

Arguments

- `dataMatrix`: Dataset containing the dependent and independent variables
- `dv`: The dependent variable named in the dataset
- `ivlist`: List of independent variables named in the dataset

Details

Function returns all possible subset information that is used by `commonality` and `dominance`. If data are missing, non-missing data are eliminated based on listwise deletion for full model.

Value

- `ivID`: Matrix containing independent variable IDS.
- `PredBitMap`: All possible subsets predictor bit map.
- `apsBitMap`: Index into all possible subsets predictor bit map.
- `APSMatrix`: Table containing the number of predictors and Multiple $R^2$ for each possible set of predictors.

Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


See Also

`calc.yhat`, `commonality`, `dominance`, `rlw`
Examples

```r
## APS regression predicting miles per gallon based
## on vehicle weight, type of
## carborator, & number of engine cylinders
apsOut<-aps(mtcars,"mpg",list("wt","carb","cyl"))

## APS regression predicting paragraph comprehension based
## on three verbal tests: general info, sentence comprehension,
## & word classification

## Use HS dataset in MBESS
if (require ("MBESS")){
data(HS)
## APS
apsOut<-aps(HS,"t6_paragraph_comprehension",list("t5_general_information","t7_sentence",
                                          "t8_word_classification"))
}
```

```
boot.yhat

Bootstrap metrics produced from /codecalc.yhat
```

Description

This function is input to `boot` to bootstrap metrics computed from `calc.yhat`.

Usage

```r
boot.yhat(data, indices, lmOut, regrout0)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Original dataset</td>
</tr>
<tr>
<td>indices</td>
<td>Vector of indices which define the bootstrap sample</td>
</tr>
<tr>
<td>lmOut</td>
<td>Output of /codelm</td>
</tr>
<tr>
<td>regrout0</td>
<td>Output of /codecalc.yhat</td>
</tr>
</tbody>
</table>

Details

This function is input to `boot` to bootstrap metrics computed from `calc.yhat`.

Value

The output of `boot.yhat` when used in conjunction with `boot` is of class `boot` and is not further described here. The output is designed to be useful as input for `booteval.yhat`

Author(s)

Kim Nimon <kim.nimon@gmail.com>
References


See Also

`lm`, `calc.yhat`, `boot`, `booteval.yhat`

Examples

```r
## Bootstrap regression results predicting paragraph comprehension based on three verbal tests: general info, sentence comprehension, & word classification

## Use HS dataset in MBESS
if (require("MBESS")){
data(HS)

## Regression
lm.out<-lm(t6_paragraph_comprehension~t5_general_information+t7_sentence+t8_word_classification, data=HS)

## Calculate regression metrics
regrOut<-calc.yhat(lm.out)

## Bootstrap results
require("boot")
boot.out<-boot(HS,boot.yhat,100,lmOut=lm.out,regrout0=regrOut)
}
```

booteval.yhat

Evaluate bootstrap metrics produced from `calc.yhat`

Description

This function evaluates the bootstrap metrics produced from `boot.yhat`.

Usage

```r
booteval.yhat(regrOut, boot.out, bty, level, prec)
```

Arguments

- `regrOut`: Output from `calc.yhat`
- `boot.out`: Output from `boot` in conjunction with `boot.yhat`
- `bty`: Type of confidence interval. Only types "perc", "norm", "basic", and "bca" supported.
- `level`: Confidence level (e.g., .95)
- `prec`: Integer indicating number of decimal places to be used.
The function evaluates the bootstrap metrics produced from `boot.yhat`. The function also outputs means, standard errors, probabilities, and reproducibility metrics for the dominance comparisons. Means and standard deviations are reported for Kendall's tau correlation between sample predictor metrics and the bootstrap statistics of like metrics.

```
combCIpm       Upper and lower CIs for predictor metrics
lowerCIpm      Lower CIs for predictor metrics
upperCIpm      Upper CIs for predictor metrics
combCIaps      Upper and lower CIs for APS metrics
lowerCIaps     Lower CIs for APS metrics
upperCIaps     Upper CIs for APS metrics
domBoot       Dominance analysis bootstrap results
tauDS         Descriptive statistics for Kendall's tau
combCIpmDiff   Upper and lower CIs for differences between predictor metrics
lowerCIpmDiff  Lower CIs for differences between predictor metrics
upperCIpmDiff  Upper CIs for differences between predictor metrics
combCIapsDiff  Upper and lower CIs for differences between APS metrics
lowerCIapsDiff Lower CIs for differences between APS metrics
upperCIapsDiff Upper CIs for differences between APS metrics
combCIincDiff  Upper and lower CIs for differences between incremental validity metrics
lowerCIincDiff Lower CIs for differences between incremental validity metrics
upperCIincDiff Upper CIs for differences between incremental validity metrics
```

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>

**References**


**See Also**

`lm calc.yhat boot plotCI.yhat`
Examples

```r
## Bootstrap regression results predicting paragraph
## comprehension based on four verbal tests: general info,
## sentence comprehension, & word classification

## Use HS dataset in MBESS
if (require ("MBESS")){
  data(HS)

  ## Regression
  lm.out<-lm(t6_paragraph_comprehension~
            t5_general_information+t7_sentence+t8_word_classification,data=HS)

  ## Calculate regression metrics
  regrOut<-calc.yhat(lm.out)

  ## Bootstrap results
  require ("boot")
  boot.out<-boot(HS,boot.yhat,100,lmOut=lm.out,regrout0=regrOut)

  ## Evaluate bootstrap results
  result<-booteval.yhat(regrOut,boot.out,bty="perc")
}
```

Description

Reports beta weights, validity coefficients, structure coefficients, product measures, commonality analysis coefficients, and dominance analysis weights for `lm` class objects.

Usage

`calc.yhat(lm.out, prec=3)`

Arguments

- `lm.out` : `lm` class object
- `prec` : level of precision for rounding, defaults to 3

Details

Takes the `lm` class object and reports beta weights, validity coefficients, structure coefficients, product measures, commonality analysis coefficients, and dominance analysis weights.
canonCommonality

Value

- PredictorMetrics: Predictor metrics associated with lm class object
- OrderedPredictorMetrics: Rank order of predictor metrics
- PairedDominanceMetrics: Dominance analysis for predictor pairs
- APSRelatedMetrics: APS metrics associated with lm class object

Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


Examples

```r
## Predict paragraph comprehension based on three verbal
## tests: general info, sentence comprehension, & word
## classification

## Use HS dataset in MBESS
if (require("MBESS")){
  data(HS)

## Regression
lm.out<-lm(t6_paragraph_comprehension~
t5_general_information+t7_sentence+t8_word_classification,data=HS)

## Regression Indices
regr.out<-calc.yhat(lm.out)
}
```

canonCommonality  Commonality Coefficients for Canonical Correlation

Description

The canonCommonality function produces commonality data for both canonical variables sets. Variables in a given canonical set are used to partition the variance of the canonical variates produced from the other canonical set and vica versa. Commonality data is supplied for the number of canonical functions requested.
Usage

canonCommonality(A, B, nofns = 1)

Arguments

A         Matrix containing variable set A
B         Matrix containing variable set B
nofns     Number of canonical functions to analyze

Details

The function canonCommonality has two required arguments and one optional argument. The first two arguments contain the two variable sets. The third argument is optional and defines the number of canonical functions to analyze. Unless specified, the number of canonical functions defaults to 1.

The function canonCommonality calls a function canonVariate to decompose canonical variates twice: the first time for the variable set identified in the first argument, the second time for the variable set identified in the second argument.

Value

The function canonCommonality returns commonality data for both canonical variable sets. For the number of functions requested, both canonical variates are analyzed. For each canonical variate analyzed, two tables are returned. The first table lists the commonality coefficients and their contribution to the total effect, while the second table lists the unique and common effects for each regressor. The function returns the resulting output ordering the output according to the function’s parameters.

Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


See Also

canonVariate

Examples

```r
## Example parallels the R builtin cancor and the
## yacca cca example
data(LifeCycleSavings)
pop <- LifeCycleSavings[, 2:3]
oec <- LifeCycleSavings[, -(2:3)]
```
## Perform Commonality Coefficient Analysis

canonCommonData<-canonCommonality(pop, oec, 1)

## Use HS dataset in MBESS
if (require("MBESS")){
data(HS)
attach(HS)

## Create canonical variable sets
MATH_REASON<-HS[,c("t20_deduction","t22_problem_reasoning")]
MATH_FUND<-HS[,c("t21_numerical_puzzles","t24_woody_mccall","t10_addition")]

## Perform Commonality Coefficient Analysis
canonCommonData<-canonCommonality(MATH_FUND, MATH_REASON, 1)
detach(HS)
}

---

canonVariate | Canonical Commonality Analysis

### Description

The `canonCommonality` function produces commonality data for a given canonical variable set. Using the variables in a given canonical set to partition the variance of the canonical variates produced from the *other* canonical set, commonality data is supplied for the number of canonical functions requested.

### Usage

canonVariate(A, B, nofns)

### Arguments

- **A**: Matrix containing variable set A
- **B**: Matrix containing variable set B
- **nofns**: Number of canonical functions to analyze

### Details

For each canonical function, `canonVariate` creates a dataset that combines the matrix of variables for a given canonical set and the canonical variate for the *other* canonical set; it calls `commonalityCoefficients`, passing the dataset, the name of the canonical variate, and the names of the variates in a given canonical set; saves resultant output.

### Value

The function `canonVariate` returns commonality data for the canonical variable set input. For the number of functions requested, two tables are returned. The first table lists the commonality coefficients for each canonical function together with its contribution to the total effect, while the second table lists the unique and common effects for each regressor.
ci.yhat

Note
This function is internal to canonCommonality, called during runtime and passed the appropriate parameters. This is not an end-user function.

Author(s)
Kim Nimon <kim.nimon@gmail.com>

References

See Also
canonCommonality

---

Description
This function retrieves the proper elements from boot.ci.

Usage
ci.yhat(bty, CI)

Arguments
bty Type of CI
CI CI

Details
This function retrieves the proper elements from boot.ci.

Value
This function returns the proper elements from boot.ci.

Note
This function is internal to the yhat package and not intended to be an end-user function.
Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


---

### combCI

*Combine upper and lower confidence intervals*

---

**Description**

This function combines upper and lower confidence intervals along with sample statistics and optionally stars intervals that do not contain 0.

**Usage**

```r
combCI(lowerCI, upperCI, est, star=FALSE)
```

**Arguments**

- `lowerCI`: Lower CI
- `upperCI`: Upper CI
- `est`: Estimate
- `star`: Boolean to indicate whether CIs that do not contain zero should be starred.

**Details**

This function evaluates the bootstrap metrics produced from `codeboot.yhat`.

**Value**

Returns estimate with confidence interval in ( ). Optionally, confidence interval not containing 0 is starred.

**Note**

This function is internal to the yhat package and not intended to be an end-user function.

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>

**References**

commonality

Commonality Analysis

Description
This function conducts commonality analyses based on an all-possible-subsets regression.

Usage
```
commonality(apsOut)
```

Arguments
```
apsOut Output from /codeaps
```

Details
This function conducts commonality analyses based on an all-possible-subsets regression.

Value
The function returns a matrix containing commonality coefficients and percentage of regression effect for each each possible set of predictors.

Author(s)
Kim Nimon <kim.nimon@gmail.com>

References

See Also
```
aps calc.yhat dominance rlw
```

Examples
```
## Predict paragraph comprehension based on three verbal
tests: general info, sentence comprehension, & word
classification
## Use HS dataset in MBESS
if (require ("MBESS")){
```
data(HS)

## All-possible-subsets regression
apsOut=aps(HS,"t6_paragraph_comprehension",
list("t5_general_information","t7_sentence","t8_word_classification"))

## Commonality analysis
commonality(apsOut)

---

**commonalityCoefficients**

*Commonality Coefficients*

**Description**

Commonality Coefficients returns a list of two tables. The first table **CC** contains the list of commonality coefficients and the percent variance for each effect. The second **CCTotByVar** totals the unique and common effects for each independent variable.

**Usage**

`commonalityCoefficients(dataMatrix, dv, ivlist, imat=FALSE)`

**Arguments**

- `dataMatrix`: Dataset containing the dependent and independent variables
- `dv`: The dependent variable named in the dataset
- `ivlist`: List of independent variables named in the dataset
- `imat`: Echo flag, default to FALSE

**Details**

When echo flag is true, transitional matrices during commonality coefficient calculation are sent to output window. Default for this option is false. When set to true, the intermediate matrices for each commonality coefficient and regression combinations are printed in the output window.

**Value**

- **CC**: Matrix containing commonality coefficients and percentage of variance for each effect.
- **CCTotalByVar**: Table of unique and common effects for each independent variable.

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>
References


See Also

canonCommonality genList odd setBits

Examples

## Predict miles per gallon based on vehicle weight, type of
carborator, & number of engine cylinders
commonalityCoefficients(mtcars,"mpg",list("wt","carb","cyl"))

## Predict paragraph comprehension based on four verbal
tests: general info, sentence comprehension, word
classification, & word type
## Use HS dataset in MBESS
if (require ("MBESS")){
data(HS)
## Commonality Coefficient Analysis
commonalityCoefficients(HS,"t6_paragraph_comprehension",list("t5_general_information","t7_sentence","t8_word_classification","t9_word_meaning"))}

---

dombin                         Dominance Analysis

Description

For each level of dominance and pairs of predictors in the full model, this function indicates whether a predictor "x1" dominates "x2", predictor "x2" dominates "x1", or that dominance cannot be established between predictors.

Usage

dombin(domOut)

Arguments

domOut          Output from /codedominance

Details

For each level of dominance and pairs of predictors in the full model, this function indicates whether a predictor "x1" dominates "x2", predictor "x2" dominates "x1", or that dominance cannot be established between predictors.
Value

The function return a matrix that contains dominance level decisions (complete, conditional, and general) for each pair of predictors in the full model.

Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


See Also

`aps`, `calc.yhat`, `commonality`, `dominance`, `rlw`

Examples

```r
## Predict paragraph comprehension based on three verbal
tests: general info, sentence comprehension, & word
classification

## Use HS dataset in MBESS
if (require("MBESS")){
data(HS)

## All-possible-subsets regression
apsOut=aps(HS,"t6_paragraph_comprehension",
list("t5_general_information", "t7_sentence","t8_word_classification"))

## Dominance analysis
domOut=dominance(apsOut)

## Dominance analysis
dombin(domOut)
}
```

---

**dominance**

*Dominance Weights*

Description

Computes dominance weights including conditional and general.

Usage

dominance(apsOut)
Arguments

apsOut    Output from /codeaps

Details

Provides full dominance weights table that are used to compute conditional and general dominance weights as well as reports conditional and general dominance weights.

Value

DA        Dominance analysis table
CD        Conditional dominance weights
GD        General dominance weights

Author(s)

Kim Nimon <kim.nimon@gmail.com>

References


See Also

aps calc.yhat dombin rlw

Examples

```r
## Predict paragraph comprehension based on three verbal
## tests: general info, sentence comprehension, & word
## classification

## Use HS dataset in MBESS
if (require ("MBESS")){
data(HS)

## All-possible-subsets regression
apsOut=aps(HS,"t6_paragraph_comprehension",
       list("t5_general_information", "t7_sentence","t8_word_classification"))

## Dominance weights
dominance(apsOut)
}
```
effect.size

Effect Size Computation for lm

Description

Creates adjusted effect sizes for linear regression.

Usage

effect.size(lm.out)

Arguments

lm.out Output from lm class object

Details

The function effect.size produces a family of effect size corrections for the R-squared metric produced from an lm class object. Suggestions for recommended correction are supplied, based on Yin and Fan (2001).

Value

Returns adjusted R-squared metric.

Author(s)

J. Kyle Roberts <kyler@smu.edu>

References


See Also

regr.yhat

Examples

if (require("MBESS")){
data(HS)
attach(HS)
lm.out<-lm(t20_deduction~t10_addition*t24_woody_mccall)
effect.size(lm.out)
detach(HS)
}
**genList**  
*Generate List $R^2$ Values*

**Description**  
Use the bitmap matrix to generate the list of $R^2$ values needed.

**Usage**  
`genList(ivlist, value)`

**Arguments**  
- `ivlist`: List of independent variables in dataset  
- `value`: Number of variables

**Details**  
Returns the number of $R^2$ values that will be calculated in output tables.

**Value**  
Returns `newlist` from generate list function call.

**Note**  
This function is internal to `commonalityCoefficients`, called during runtime and passed the appropriate parameters. This is not an end-user function.

**Author(s)**  
Kim Nimon <kim.nimon@gmail.com>

---

**odd**  
*isOdd Function*

**Description**  
Function receives value and returns true if value is odd.

**Usage**  
`odd(val)`

**Arguments**  
- `val`: Value to check
plotCI.yhat

Details
Determines value of parameter in argument.

Value
Returns true when value checked is odd. Otherwise, function returns a value false.

Note
This function is internal to commonalityCoefficients, called during runtime and passed the appropriate parameters. This is not an end-user function.

Author(s)
Kim Nimon <kim.nimon@gmail.com>

---

Description
This function plots CIs that have been produced from /codebooteval.yhat.

Usage
plotCI.yhat(sampStat, upperCI, lowerCI, pid=1:ncol(sampStat), nr=2, nc=2)

Arguments
- sampStat: Set of sample statistics
- upperCI: Set of upper CIs
- lowerCI: Set of lower CIs
- pid: Which set of Metrics to plot (default to all)
- nr: Number of rows (default = 2)
- nc: Number of columns (default = 2)

Details
This function plots CIs that have been produced from /codebooteval.yhat.

Value
This returns a plot of CIs that have been produced from /codebooteval.yhat.

Author(s)
Kim Nimon <kim.nimon@gmail.com>
References


See Also

`lm`, `calc.yhat`, `boot`, `booteval.yhat`

Examples

```r
## Bootstrap regression results predicting paragraph
## comprehension based on three verbal tests: general info, sentence comprehension, & word classification

## Use HS dataset in MBESS
if (require("MBESS")){
  data(HS)

## Regression
lm.out<-lm(t6_paragraph_comprehension~t5_general_information+t7_sentence+t8_word_classification,data=HS)

## Calculate regression metrics
regrOut<-calc.yhat(lm.out)

## Bootstrap results
require("boot")
boot.out<-boot(HS,boot.yhat,100,lmOut=lm.out,regrout0=regrOut)

## Evaluate bootstrap results
result<-booteval.yhat(regrOut,boot.out,bty="perc")

## Plot results
## plotCI.yhat(regrOut$PredictorMetrics[-nrow(regrOut$PredictorMetrics),],
## result$upperCIpm,result$lowerCIpm, pid=which(colnames(regrOut$PredictorMetrics)
## %in% c("Beta","rs","CD:0","CD:1","CD:2","GenDom","Pratt","RLW") == TRUE),nr=3,nc=3)
```

**regr**  
*Regression effect reporting for lm class objects*

Description

The *regr* reports beta weights, standardized beta weights, structure coefficients, adjusted effect sizes, and commonality coefficients for *lm* class objects.

Usage

`regr(lm.out)`
Arguments

lm.out lm class object

Details

The function `regr` takes the lm class object and reports beta weights, standardized beta weights, structure coefficients, adjusted effect sizes, and commonality coefficients for lm class objects.

Value

LM_Output The summary of the output from the lm class object
Beta_Weights Beta weights for the regression effects
Structure_Coefficients Structure coefficients for the regression effects
Commonality_Data Commonality coefficients for the regression effects. The output only produces a parsed version of CCdata
Effect_Size Adjusted effect size computations based on R^2 adjustments

Author(s)

J. Kyle Roberts <kyler@smu.edu>, Kim Nimon <kim.nimon@gmail.com>

References


See Also

`commonalityCoefficients, effect.size`

Examples

```r
if (require("MBESS")){
  data(HS)
  attach(HS)
  lm.out<-lm(t20_deduction~t10_addition*t24_woody_mccall)
  regr(lm.out)
  detach(HS)
}
```
**Description**

The function computes relative weights.

**Usage**

```r
rlw(dataMatrix, dv, ivlist)
```

**Arguments**

- `dataMatrix`: Dataset containing the dependent and independent variables
- `dv`: The dependent variable named in the dataset
- `ivlist`: List of independent variables named in the dataset

**Details**

The function computes relative weights.

**Value**

The function returns relative weights for each predictor.

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>

**References**


**See Also**

`aps`, `calc.yhat`, `commonality`, `dominance`
Examples

## Relative weights from regression model predicting paragraph comprehension based on three verbal tests: general info, sentence comprehension, & word classification

## Use HS dataset in MBESS

```r
if (require ("MBESS")){
  data(HS)

  ## Relative Weights
  rwlOut<-rlw(HS,"t6_paragraph_comprehension",
              c("t5_general_information","t7_sentence","t8_word_classification"))
}
```

---

**setBits**  
**Decimal to Binary**

**Description**

Creates the binary representation of n and stores it in the nth column of the matrix.

**Usage**

```r
setBits(col, effectBitMap)
```

**Arguments**

- `col`: Column of matrix to represent in binary image
- `effectBitMap`: Matrix of mean combinations in binary form

**Details**

Creates the binary representation of `col` and stores it in its associated column.

**Value**

Returns matrix `effectBitMap` of mean combinations in binary form.

**Note**

This function is internal to `commonalityCoefficients`, called during runtime and passed the appropriate parameters. This is not an end-user function.

**Author(s)**

Kim Nimon <kim.nimon@gmail.com>
## Index

**models**
- `aps`, 3
- `boot.yhat`, 4
- `booteval.yhat`, 5
- `commonality`, 13
- `commonalityCoefficients`, 14
- `dombin`, 15
- `dominance`, 16
- `plotCI.yhat`, 20
- `rlw`, 23

**multivariate**
- `canonCommonality`, 8
- `canonVariate`, 10

**package**
- `yhat-package`, 2

**regression**
- `aps`, 3
- `boot.yhat`, 4
- `booteval.yhat`, 5
- `commonality`, 13
- `commonalityCoefficients`, 14
- `dombin`, 15
- `dominance`, 16
- `plotCI.yhat`, 20
- `rlw`, 23

- `aps`, 2, 3, 13, 16, 17, 23
- `boot`, 5, 6, 21
- `boot.yhat`, 2, 4
- `booteval.yhat`, 2, 5, 5, 21
- `calc.yhat`, 2, 3, 5, 6, 7, 13, 16, 17, 21, 23
- `canonCommonality`, 2, 8, 11, 15
- `canonVariate`, 9, 10
- `ci.yhat`, 11
- `combCI`, 12
- `commonality`, 2, 3, 13, 16, 23
- `commonalityCoefficients`, 2, 14, 19, 20, 22, 24
- `dombin`, 2, 15, 17
- `dominance`, 2, 3, 13, 16, 16, 23
- `effect.size`, 18, 22
- `genList`, 15, 19
- `lm`, 5, 6, 21
- `odd`, 15, 19
- `plotCI.yhat`, 2, 6, 20
- `regr`, 2, 18, 21
- `rlw`, 2, 3, 13, 16, 17, 23
- `setBits`, 15, 24
- `yhat`, 18
- `yhat (yhat-package)`, 2
- `yhat-package`, 2

25