Package 'ScottKnottESD'

January 20, 2025

Type Package

Title The Scott-Knott Effect Size Difference (ESD) Test

Version 2.0.3

Date 2018-05-08

Author Chakkrit Tantithamthavorn

Maintainer Chakkrit Tantithamthavorn <kla@chakkrit.com>

Description

The Scott-Knott Effect Size Difference (ESD) test is a mean comparison approach that leverages a hierarchical clustering to partition the set of treatment means (e.g., means of variable importance scores, means of model performance) into statistically distinct groups with non-negligible difference [Tantithamthavorn et al., (2018) <doi:10.1109/TSE.2018.2794977>].

License GPL (>= 2)

Depends reshape2, effsize, stats, car

Imports forecast

LazyData true

URL https://github.com/klainfo/ScottKnottESD

BugReports https://github.com/klainfo/ScottKnottESD/issues

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-08 07:48:19 UTC

Contents

`ScottKnottESD-package` *The Scott-Knott Effect Size Difference (ESD) Test*

Description

The Scott-Knott Effect Size Difference (ESD) test is a mean comparison approach that leverages a hierarchical clustering to partition the set of treatment means (e.g., means of variable importance scores, means of model performance) into statistically distinct groups with non-negligible difference [Tantithamthavorn et al., (2018) <doi:10.1109/TSE.2018.2794977>]. It is an alternative approach of the Scott-Knott test that considers the magnitude of the difference (i.e., effect size) of treatment means with-in a group and between groups. Therefore, the Scott-Knott ESD test (v2.x) produces the ranking of treatment means while ensuring that (1) the magnitude of the difference for all of the treatments in each group is negligible; and (2) the magnitude of the difference of treatments between groups is non-negligible.

The mechanism of the Scott-Knott ESD test (v2.x) is made up of 2 steps:

(Step 1) Find a partition that maximizes treatment means between groups. We begin by sorting the treatment means. Then, following the original Scott-Knott test, we compute the sum of squares between groups (i.e., a dispersion measure of data points) to identify a partition that maximizes treatment means between groups.

(Step 2) Splitting into two groups or merging into one group. Instead of using a likelihood ratio test and a Chi-square distribution as a splitting and merging criterion (i.e., a hypothesis testing of the equality of all treatment means), we analyze the magnitude of the difference for each pair for all of the treatment means of the two groups. If there is any one pair of treatment means of two groups are non-negligible, we split into two groups. Otherwise, we merge into one group. We use the Cohen effect size — an effect size estimate based on the difference between the two means divided by the standard deviation of the two treatment means (d = (mean(x_1) - mean(x_2))/s.d.).

Unlike the earlier version of the Scott-Knott ESD test (v1.x) that post-processes the groups that are produced by the Scott-Knott test, the Scott-Knott ESD test (v2.x) pre-processes the groups by merging pairs of statistically distinct groups that have a negligible difference.

Details

Package:	ScottKnottESD
Type:	Package
Version:	2.0.3
Date:	2017-07-03
License:	GPL (>= 2)

Author(s)

Chakkrit (Kla) Tantithamthavorn

Maintainer: Chakkrit (Kla) Tantithamthavorn <kla@chakkrit.com>

References

Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E. Hassan, Kenichi Matsumoto, An Empirical Comparison of Model Validation Techniques for Defect Prediction Models. IEEE Transactions on Software Engineering. 43(1): 1-18 (2017). <doi:10.1109/TSE.2016.2584050>

Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E. Hassan, Kenichi Matsumoto, The Impact of Automated Parameter Optimization for Defect Prediction Models. IEEE Transactions on Software Engineering. Early Access. (2018). <doi:10.1109/TSE.2018.2794977>

See Also

-

Examples

```
library(ScottKnottESD)

sk <- sk_esd(example)
plot(sk)

sk <- sk_esd(maven)
plot(sk)
```

"check.ANOVA.assumptions"

Check basic ANOVA assumptions

Description

Check the normality assumption of the input dataset using the Kolmogorov-Smirnov Test and the homogeneity of variances assumption of the input dataset using the Levene's test.

Usage

```
check.ANOVA.assumptions(x, alpha = 0.05, ...)
```

Arguments

x	A wide-format data frame.
alpha	The significance level.
...	Optional parameters.

Value

A wide-format data frame.

Author(s)

Chakkrit Tantithamthavorn (kla@chakkrit.com)

Examples

```
check.ANOVA.assumptions(example)
```

"long2wide" *Convert data from long format to wide format*

Description

Convert data from long format to wide format

Usage

```
long2wide(x, ...)
```

Arguments

x	A long-format data frame.
...	Optional parameters.

Value

A wide-format data frame.

Author(s)

Chakkrit Tantithamthavorn (kla@chakkrit.com)

Examples

```
long2wide(melt(example, id.vars=0))
```

"normalize" *Normalize non-normal distributions using the Box-Cox Power Transformation*

Description

Normalize non-normal distributions using the Box-Cox Power Transformation

Usage

```
normalize(x, ...)
```

Arguments

x	A wide-format data frame.
...	Optional parameters.

Value

A wide-format data frame.

Author(s)

Chakkrit Tantithamthavorn (kla@chakkrit.com)

Examples

```
normalized.data <- normalize(example)
```

---

example *An example dataset of Breiman's variable importance scores*

Description

A dataset containing software metrics of 1,000 calculation of Breiman's variable importance scores

Usage

```
example
```

Format

A data frame with 1,000 rows and 9 variables:

LOC lines of code

Components the numbers of components

Subsystem the numbers of subsystems

Files the numbers of files

Commit the numbers of commits

Churn the numbers of churns

Ownership ownership

Authorship authorship

Experience developer's experience ...

Source

https://github.com/klainfo/ScottKnottESD/

maven *An example dataset of Breiman's variable importance scores*

Description

A dataset containing software metrics of 1,000 calculation of Breiman's variable importance scores

Usage

```
maven
```

Format

A data frame with 1,000 rows and 27 variables:

Avg_CloneLineCount An average physical lines of clone siblings of a clone.

Avg_CountLineComment An average comment lines in the methods that contain clone siblings of a clone.

Avg_Cyclomatic McCabe Cyclomatic complexity of the method that contains the clone.

Avg_ImproveCommitCount Number of commits that impact the method containing the clone.

Avg_LineAdded Number of lines added into the method that contains the clone.

Avg_LineCodeCount Number of source code lines in the method that contains the clone.

Avg_MaxNesting Maximum nesting level of control constructs in the method that contains the clone.

Avg_NewFeatureCommitCount Number of commits that introduce new feature and that impact the method containing the clone.

Avg_RatioCommentToCode Ratio of CommentLineCount to LineCodeCount.

Avg_RatioLineCodeCount Ratio of LineCount to CloneLineCount.

Avg_TokenCount Number of tokens in the clone.

CloneType Type of clone class to which the clone belongs.

Diff_CloneLineCount Number of physical lines in the clone.

Diff_CountLineComment Number of comment lines in the method that contains the clone.

Diff_Cyclomatic McCabe Cyclomatic complexity of the method that contains the clone.

Diff_DeveloperCount Number of distinct developers who modified the method that contains the clone.

Diff_Essential Numberical measure of structuredness of the method that contains the clone.

Diff_FanIn Number of unique methods that call the method containg the clone.

Diff_FanOut Number of unique methods that are called by the method containing the clone.

Diff_FixCommitCount Number of commits with a description of fixing bugs and that impact the method containing the clone.

Diff_LineCodeDeclCount Number of declarative source code lines in the method that contains the clone.

Diff_LineCount Number of lines in the method that contains the clone.

Diff_LineDeleted Number of lines deleted from the method that contains the clone.

Diff_NewFeatureCommitCount Number of commits that introduce new feature and that impact the method containing the clone.

Diff_TokenCount Number of tokens in the clone.

Max_DirectoryDistance Number of directories that are traversed from the method containing one sibling to the method containing another sibling of the clone.

SiblingCount Number of clone siblings in the clone.

Source

https://github.com/klainfo/ScottKnottESD/

`print.sk_esd` *Print sk_esd objects*

Description

S3 method to print sk_esd objects.

Usage

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

Arguments

`x`	A sk_esd object
`...`	Optional parameters.

Value

The sk.esd ranks

sk_esd *A function to check the magnitude of the difference for all pairs of treatments*

Description

A function to check the magnitude of the difference for all pairs of treatments

An enhancement of the Scott-Knott test (which cluster distributions into statistically distinct ranks) that takes effect size into consideration.

Usage

```
checkDifference(ranking, data)

sk_esd(x, alpha = 0.05, ...)
```

Arguments

ranking	A ranking that is produced by the Scott-Knott ESD test
data	a data frame of treatment means
x	A wide-format data frame.
alpha	The significance level.
...	Optional parameters.

Value

A result of the magnitude of the difference for all pairs of treatments.

A sk_esd object.

Author(s)

Chakkrit Tantithamthavorn (kla@chakkrit.com)

Examples

```
sk <- sk_esd(example)
checkDifference(sk$groups, example)

sk <- sk_esd(example)
plot(sk)

sk <- sk_esd(maven)
plot(sk)
```

Index