The material contained in this Documentation is in part also contained in the function `example.ci` of the package.

1 Main and interaction effect analysis

Organise your data in a `data.frame` X. Eg.:

```
instance algorithm class results
inst1    alg1      boh1   40
inst1    alg2      boh1   40
...
inst1    alg1      boh2   40
inst1    alg2      boh2   40
...
```

`instance, algorithm` and `class` must be factors. Note that the columns names can be redefined but `class` must remain.

```
> formula <- results ~ instance * algorithm
```

See R documentation for formula syntax (above all on the difference between the sign "*" and "+" in the formula).

Parametric analysis:

```
> aov(formula,data=X)
```

No need for the library so far. Everything is already in R.

Then load the library:

```
> source("lib_effects.R")
```

Nonparametric analysis (by ranks):

```
>  Friedman.test(formula,data=X)
```

Nonparametric permutation analysis :

```
> permut.aov(formula,data=X,method="same",B=IN.PERM,type="csp")
```

Values `eff1 eff2 inter` represent main inteeractions and intereaction effects, respectively.

2 Post-hoc analysis

The packages `lattice` e `Hmisc` must be installed.
Organise your data in a `data.frame` X. Eg.:

```
instance algorithm class results
inst1    alg1      boh1   40
inst1    alg2      boh1   40
...
inst1    alg1      boh2   40
inst1    alg2      boh2   40
...
```

instance, algorithm and class must be factors.
Then from R, load the library:

```
> source("lib_posthoc.R")
```

for a parametric analysis:

```
> R <-AP.SCI(results~algorithm*instance,data=X,method="Parametric",test="Tukey")
> plot.ci(R)
```

Note that, in the formula above `algorithm * instance` accounts also for a significant interaction while `algorithm + instance` does not account for interaction.
In the nonparametric case:

```
> R<-AP.SCI(results~algorithm+instance,data=X,method="Ranks",test="Conover")
> plot.ci(R)
```

Interaction cannot be handled in this case. The output will be in a file named `plot.ci.eps`. For improving graphical layout edit the function `plot.ci.xclass`.

If instead the analysis includes more than one class of instances that for some reason one wants to maintain distinct:

```
> R <- AP.SCI.wrapper.class(results~algorithm*instance,data=X,
                            method="Parametric",test="Tukey")
> plot.ci.xclass(R)
```

or

```
> R <- AP.SCI.wrapper.class(results~algorithm+instance,data=X,
                            method="Ranks",test="Conover")
> plot.ci.xclass(R)
```

The output will be in a file named `plot.ci.xclass.eps`. For improving graphical layout edit the function `plot.ci.xclass`.

2.1 Complete nomenclature

`formula:` ...

`method: "Parametric", "Ranks", "Permutations"`

`test:` Parametric case: `"Tukey", "LSD", "LSDBonferroni", "Sheffe"`
Nonparametric case: `"Conover","Sheskin","Permutations"`, i.e., Friedman test from Conover or Sheskin or permutation test on ranks.

`alpha:` is the family or experiment wise error rate.

`adj.method:` only in Permutation method. `"simul.Bonferroni", ‘‘none’’ "Bonferroni"`

`type: "csp","usp"` Constrained and Unconstrained Synchrnised Permutations

`B:` only in Permutation method. Numer of Monte Carlo Samples from the permutation space.

Values

`lower` and `upper`