Krippendorff's alpha

Cooper Stansbury

13 May, 2020

Contents

1 Preliminaries

1.1 Software Dependencies

Here are the packages used in this analysis.

```
library("irr")
library("boot")
library("knitr")
library("psych")
library("tidyverse")
```

1.2 Note on Confidence Intervals and Cohen's Kappa Weights

In this document condifence intervals are computed for default $\alpha = 0.05$. Contrasts weights are defaults also, see the Cohen's Kappa sections for ordinal coding details. Because there are a large number of ways to perform the weighted Cohen's Kappa, I think it's worth glancing at the following publications [1-2]. Based on this work I think it is important to choose a the correct weight matrix for Cohen's Kappa computations This can be found in Table 3 of the publication cited below. I believe that 'Dichotomous-ordinal' is the correcting weight scheme. As presented by both piublications Warrens MJ. This scheme is as follows:

0	w_3	w_2
w_3	0	w_1
w_2	w_1	0

Table 1: Dichotomous-ordinal Weights from Warrens MJ. [1-2]

Please note that this is exactly the same as the first scheme labeled 'General symmetric.' The default weights of the `pysch` package are slightly different, and most closely aligns with the 'linear' scheme in the same table, though the diagonals are different.

1	0.75	0
0.75	1	0.75
0	0.75	1

Table 2: 'psych' Default Weights

1. Warrens MJ. Weighted Kappas for Tables [Internet]. Vol. 2013, Journal of Probability and Statistics. Hindawi; 2013 [cited 2020 May 13]. p. e325831. Available from: https://www.hindawi.com/journals/jps/2013/325831/

2. Warrens MJ. Weighted Kappas for 3 × 3 Tables. Journal of Probability and Statistics. 2013;2013:1–9.

From the `pysch` package documentation:

On confidence intervals:

"The confidence intervals are based upon the variance estimates discussed by Fleiss, Cohen, and Everitt who corrected the formulae of Cohen (1968) and Blashfield."

On weights:

"Kappa considers the matches on the main diagonal. A penalty function (weight) may be applied to the off diagonal matches. If the weights increase by the square of the distance from the diagonal, weighted kappa is similar to an Intra Class Correlation (ICC). Derivations of weighted kappa are sometimes expressed in terms of similarities, and sometimes in terms of dissimilarities. In the latter case, the weights on the diagonal are 1 and the weights off the diagonal are less than one. In this case, if the weights are 1 - squared distance from the diagonal / k, then the result is similar to the ICC (for any positive k)."

1.3 Loading/Coding Annotations

For `AB.separate` the coding is as follows: - A = 3 - B = 2 - Blank = 1

For `AB.together` the coding is as follows: - AB = 2 - Blank = 1

```
AB.separate <- read.csv("AB_separate.csv")
AB.together <- read.csv("AB_together.csv")

knitr::kable(head(AB.separate),
             caption='Annotations A/B Separate')
```

Table 3: Annotations A/B Separate

KATHLEEN	LIZ	KAYCEE
Blank	Blank	Blank
Blank	Blank	A
Blank	Blank	A
Blank	Blank	Blank
Blank	Blank	Blank
Blank	Blank	Blank

```
knitr::kable(head(AB.together),
             caption='Annotations A/B Together')
```

Table 4: Annotations A/B Together

KATHLEEN	LIZ	KAYCEE
Blank	Blank	Blank
Blank	Blank	AB
Blank	Blank	AB
Blank	Blank	Blank
Blank	Blank	Blank
Blank	Blank	Blank

By default, factors are loaded in reverse order. We'll rearrange the default factor level encoding so that it matches the list above. This is important for ordinal factor Cohen's Kappa calculations.

```
# reorder factors in `separate` table
AB.separate$LIZ <- fct_rev(AB.separate$LIZ)
AB.separate$KAYCEE <- fct_rev(AB.separate$KAYCEE)
AB.separate$KATHLEEN <- fct_rev(AB.separate$KATHLEEN)
str(AB.separate)
```

```
## 'data.frame':    6399 obs. of  3 variables:
##  $ KATHLEEN: Factor w/ 3 levels "Blank","B","A": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LIZ     : Factor w/ 3 levels "Blank","B","A": 1 1 1 1 1 1 1 1 1 1 ...
##  $ KAYCEE  : Factor w/ 3 levels "Blank","B","A": 1 3 3 1 1 1 1 1 1 1 ...
```

```
# reorder factors in `together` table
AB.together$LIZ <- fct_rev(AB.together$LIZ)
AB.together$KAYCEE <- fct_rev(AB.together$KAYCEE)
AB.together$KATHLEEN <- fct_rev(AB.together$KATHLEEN)
str(AB.together)
```

```
## 'data.frame':    6399 obs. of  3 variables:
##  $ KATHLEEN: Factor w/ 2 levels "Blank","AB": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LIZ     : Factor w/ 2 levels "Blank","AB": 1 1 1 1 1 1 1 1 1 1 ...
##  $ KAYCEE  : Factor w/ 2 levels "Blank","AB": 1 2 2 1 1 1 1 1 1 1 ...
```

2 Pairwise Cohen's Kappas

2.1 Weights

We need to define weights per the note above. Please note that the weights are counter-intuitive: $w_3 = 2$ and $w_2 = 3$. I defer to the cited publications here.

```
w <- matrix(c(0, 2, 3, 2,0,1, 3,1,0), nrow = 3)
w
```

```
##      [,1] [,2] [,3]
## [1,]    0    2    3
## [2,]    2    0    1
## [3,]    3    1    0
```

2.2 Weighted and Unweighted Cohen's for A/B Separately

For each pair of annotators we calculate Cohen's Kappa (Cohen Kappa is below the diagonal and Weighted Kappa is above the diagonal). Note, we get the error message below if we do not specify the argument `w`, that is, the weights:

"No variance detected in cells 2 1No variance detected in cells 3 2At least one item had no variance. Try describe(your.data) to find the problem."

We'll dig into this below.

```
pairwise.cohens.sep <- cohen.kappa(AB.separate, w = w)

knitr::kable(pairwise.cohens.sep$cohen.kappa,
             caption="Weighted and Unweighted Pairwise Cohen's Kappa",
             digits = 5)
```

Table 5: Weighted and Unweighted Pairwise Cohen's Kappa

	KATHLEEN	LIZ	KAYCEE
KATHLEEN	1.00000	0.60400	0.57955
LIZ	0.56064	1.00000	0.65462
KAYCEE	0.54661	0.61654	1.00000

We'll also present 95% confidence intervals around the Cohen's Kappas:

```
liz.kath.ci <- pairwise.cohens.sep$`KATHLEEN LIZ`$confid
kath.kay.ci <- pairwise.cohens.sep$`KATHLEEN KAYCEE`$confid
liz.kay.ci <- pairwise.cohens.sep$`LIZ KAYCEE`$confid
```

```
knitr::kable(liz.kath.ci, caption='Liz vs. Kathleen 95% Confidence Intervals',
             digits=4)
```

Table 6: Liz vs. Kathleen 95% Confidence Intervals

	lower	estimate	upper
unweighted kappa	0.5212	0.5606	0.6001
weighted kappa	0.4152	0.6040	0.7928

```
knitr::kable(kath.kay.ci, caption='Kaycee vs. Kathleen 95% Confidence Intervals',
             digits=4)
```

Table 7: Kaycee vs. Kathleen 95% Confidence Intervals

	lower	estimate	upper
unweighted kappa	0.5078	0.5466	0.5854
weighted kappa	0.3722	0.5796	0.7869

```
knitr::kable(liz.kay.ci, caption='Liz vs. Kaycee 95% Confidence Intervals',
             digits=4)
```

Table 8: Liz vs. Kaycee 95% Confidence Intervals

	lower	estimate	upper
unweighted kappa	0.5787	0.6165	0.6544
weighted kappa	0.4586	0.6546	0.8506

For completeness, here are the intermediate agreements.

```
liz.kath.agree <- pairwise.cohens.sep$`KATHLEEN LIZ`$agree
kath.kay.agree <- pairwise.cohens.sep$`KATHLEEN KAYCEE`$agree
liz.kay.agree <- pairwise.cohens.sep$`LIZ KAYCEE`$agree

knitr::kable(liz.kath.agree, caption='Liz vs. Kaycee Agreement Percentages',
             digits=4)
```

Table 9: Liz vs. Kaycee Agreement Percentages

	A	B	Blank
A	0.0370	0.0011	0.0139
B	0.0028	0.0005	0.0059
Blank	0.0247	0.0072	0.9069

```
knitr::kable(kath.kay.agree, caption='Liz vs. Kaycee Agreement Percentages',
             digits=4)
```

Table 10: Liz vs. Kaycee Agreement Percentages

	A	B	Blank
A	0.0419	0.0011	0.0275
B	0.0005	0.0002	0.0039
Blank	0.0222	0.0075	0.8953

```
knitr::kable(liz.kay.agree, caption='Liz vs. Kaycee Agreement Percentages',
             digits=4)
```

Table 11: Liz vs. Kaycee Agreement Percentages

	A	B	Blank
A	0.0419	0.0027	0.0259
B	0.0005	0.0002	0.0039
Blank	0.0097	0.0064	0.9089

2.3 Weighted and Unweighted Cohen's for A/B Together

The weight matrix used is as shown below. Weights will no affect the computation in the binary case.

Table 12: Weights Used for Cohen's Kappa

	AB	Blank
AB	1	0
Blank	0	1

```
pairwise.cohens.tog <- cohen.kappa(AB.together)

knitr::kable(pairwise.cohens.tog$cohen.kappa,
             caption="Cohen's Kappa for A==B",
             digits=4)
```

Table 13: Cohen's Kappa for A==B

	KATHLEEN	LIZ	KAYCEE
KATHLEEN	1.0000	0.5881	0.5550
LIZ	0.5881	1.0000	0.6385
KAYCEE	0.5550	0.6385	1.0000

Here are the confidence intervals:

```
knitr::kable(pairwise.cohens.tog$`KATHLEEN LIZ`$confid,
             caption="Liz v. Kathleen Cohen's Kappa",
             digits = 4)
```

Table 14: Liz v. Kathleen Cohen's Kappa

	lower	estimate	upper
unweighted kappa	0.5479	0.5881	0.6282
weighted kappa	0.5479	0.5881	0.6282

```
knitr::kable(pairwise.cohens.tog$`LIZ KAYCEE`$confid,
             caption="Liz v. Kaycee Cohen's Kappa",
             digits = 4)
```

Table 15: Liz v. Kaycee Cohen's Kappa

	lower	estimate	upper
unweighted kappa	0.6003	0.6385	0.6767
weighted kappa	0.6003	0.6385	0.6767

```
knitr::kable(pairwise.cohens.tog$`KATHLEEN KAYCEE`$confid,
             caption="Kathleen v. Kaycee Cohen's Kappa",
             digits = 4)
```

Table 16: Kathleen v. Kaycee Cohen's Kappa

	lower	estimate	upper
unweighted kappa	0.5157	0.555	0.5943
weighted kappa	0.5157	0.555	0.5943

3 Krippendorff's alpha

This is the `kripp.alpha()` function from: https://cran.r-project.org/web/packages/irr/irr.pdf run on the data where blanks are explcitly coded (not `NA`). We provide both nominal and ordinal estimates. From the supplementary material of the foillowing publication we find this a note indicating the lack of confidence intervals for this function, as well as small estimation errors due to the lack of `NA` values. It is worth noting that similar behavior is seen in the Python package `krippendorff`.

1. Zapf A, Castell S, Morawietz L, Karch A. Measuring inter-rater reliability for nominal data – which coefficients and confidence intervals are appropriate? BMC Medical Research Methodology. 2016 Aug 5;16(1):93.

"In R (R Core Team, Vienna, Austria) there is the package irr (version 0.84) from Gamer et al. [2], which calculates Fleiss' K and Krippendorff's alpha, but both without confidence intervals. There is a small error in the estimation of the coincidence matrix for Krippendorff's alpha if there are no missing values. In the

upcoming actualized version this error will be corrected (personal communication). An R-program for the calculation of Krippendorffs alpha with the standard bootstrap confidence interval as applied by us was written by Gruszczynski and can be downloaded via GitHub [3].”

3.1 Krippendorff’s alpha A/B Separate

```
AB.sep.nominal <- kripp.alpha(t(AB.separate), "nominal")
AB.sep.nominal
```

```
##  Krippendorff's alpha
##
##  Subjects = 6399
##    Raters = 3
##     alpha = 0.574
```

```
print(paste('Full value: ', AB.sep.nominal$value))
```

```
## [1] "Full value:  0.573760698974809"
```

```
AB.sep.ordinal <- kripp.alpha(t(AB.separate), "ordinal")
AB.sep.ordinal
```

```
##  Krippendorff's alpha
##
##  Subjects = 6399
##    Raters = 3
##     alpha = 0.599
```

```
print(paste('Full value: ', AB.sep.ordinal$value))
```

```
## [1] "Full value:  0.598846330297061"
```

3.2 Krippendorff’s alpha A/B Together

Oridinal and nominal weight matrices will be identical in this case.

```
AB.together.nominal <- kripp.alpha(t(AB.together), "nominal")
AB.together.nominal
```

```
##  Krippendorff's alpha
##
##  Subjects = 6399
##    Raters = 3
##     alpha = 0.593
```

```
print(paste('Full value: ', AB.together.nominal$value))
```

```
## [1] "Full value:  0.592739012105127"
```

3.3 Booptstrapped Confidence Intervals

We'll rely on this post (https://stackoverflow.com/questions/41944703) to using bootstrapiing to build confidence intervals, since the "irr" function doesn't return them. Below is the bootstrapped confidence interval for Krippendorff's alpha:

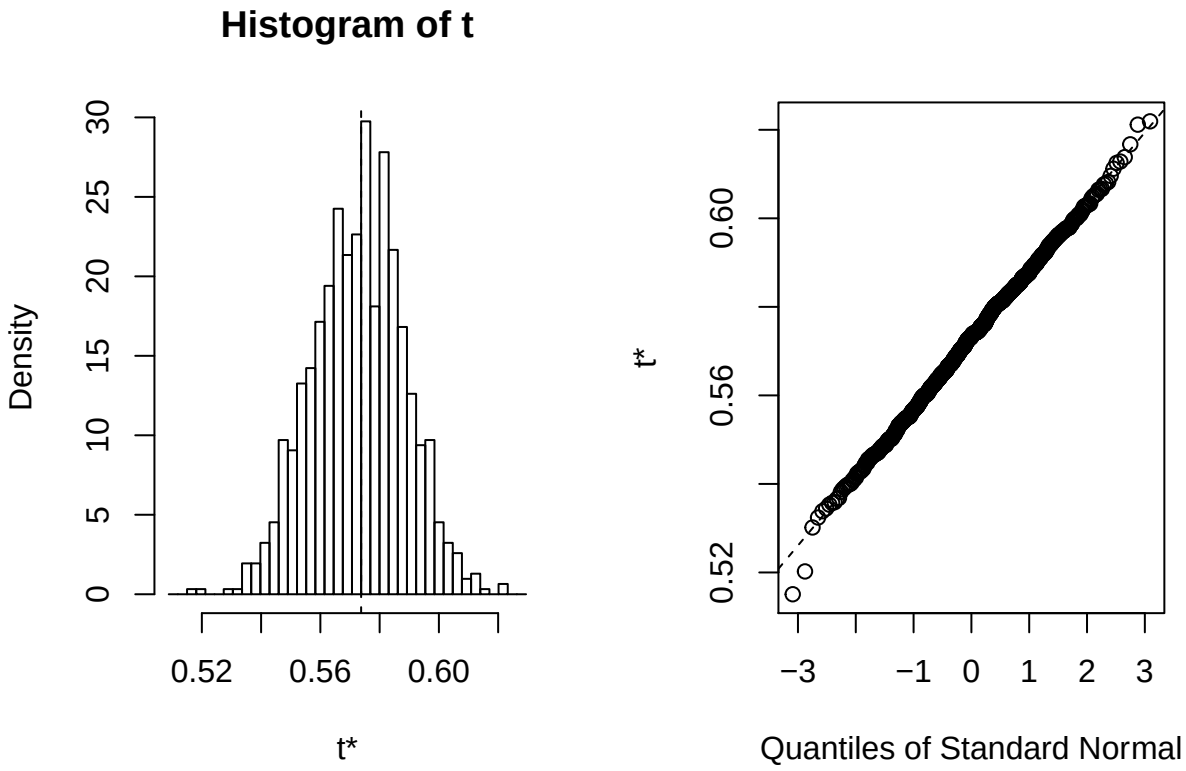
```
nominal.alpha <- function(d, w){
  #' a function bootstrap nominal coding of
  #' Krippendorff's alpha
  data <- t(d[w,])
  kripp.alpha(data, 'nominal')$value
}

oridinal.alpha <- function(d, w) {
  #' a function bootstrap nominal coding of
  #' Krippendorff's alpha
      data <- t(d[w,])
      kripp.alpha(data, 'ordinal')$value
}
```

3.4 A/B Separate, Coded Nominally

```
b <- boot(data = AB.separate, statistic = nominal.alpha, R = 1000)
b
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = AB.separate, statistic = nominal.alpha, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.5737607 -0.001022207  0.01555091
```

```
plot(b)
```




```
boot.ci(b, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.5425,  0.6028 )
## Calculations and Intervals on Original Scale
```

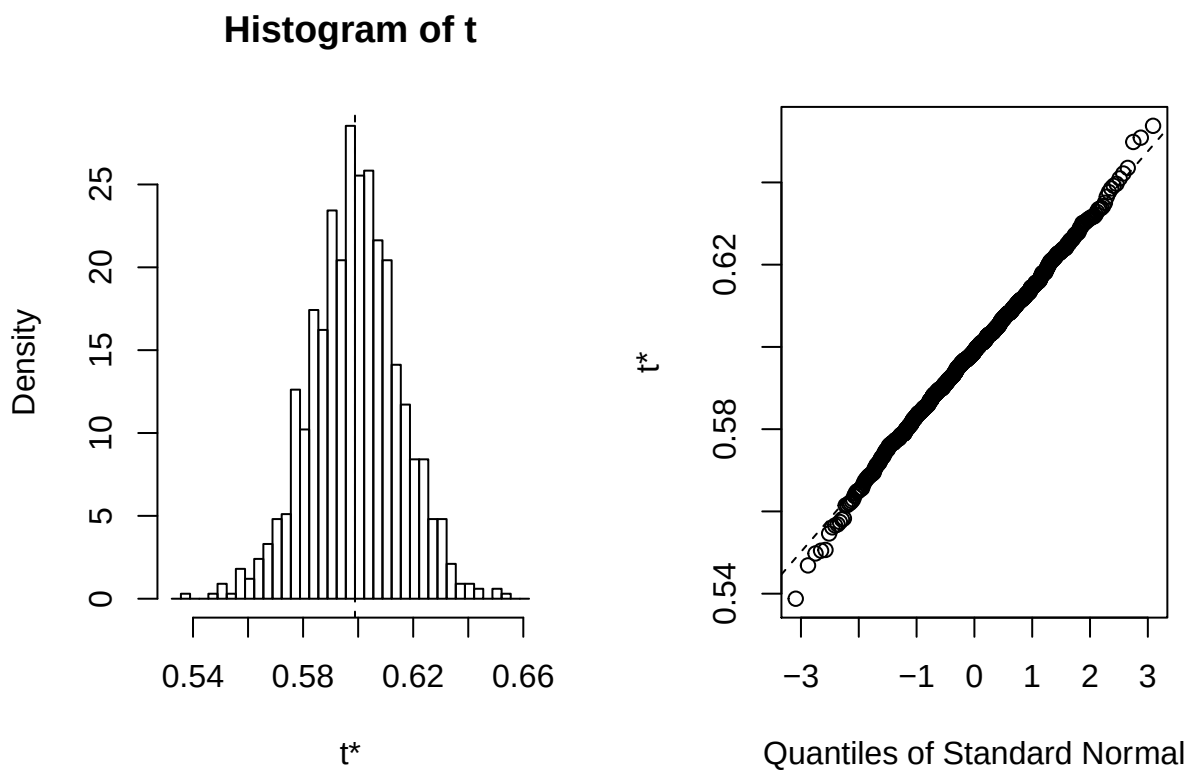
3.5 A/B Separate, Coded Ordinally

```
b <- boot(data = AB.separate, statistic = oridinal.alpha, R = 1000)
b
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
```

```
## boot(data = AB.separate, statistic = oridinal.alpha, R = 1000)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.5988463 2.07593e-05  0.01623603
```

```
plot(b)
```




```
boot.ci(b, type = "perc")
```

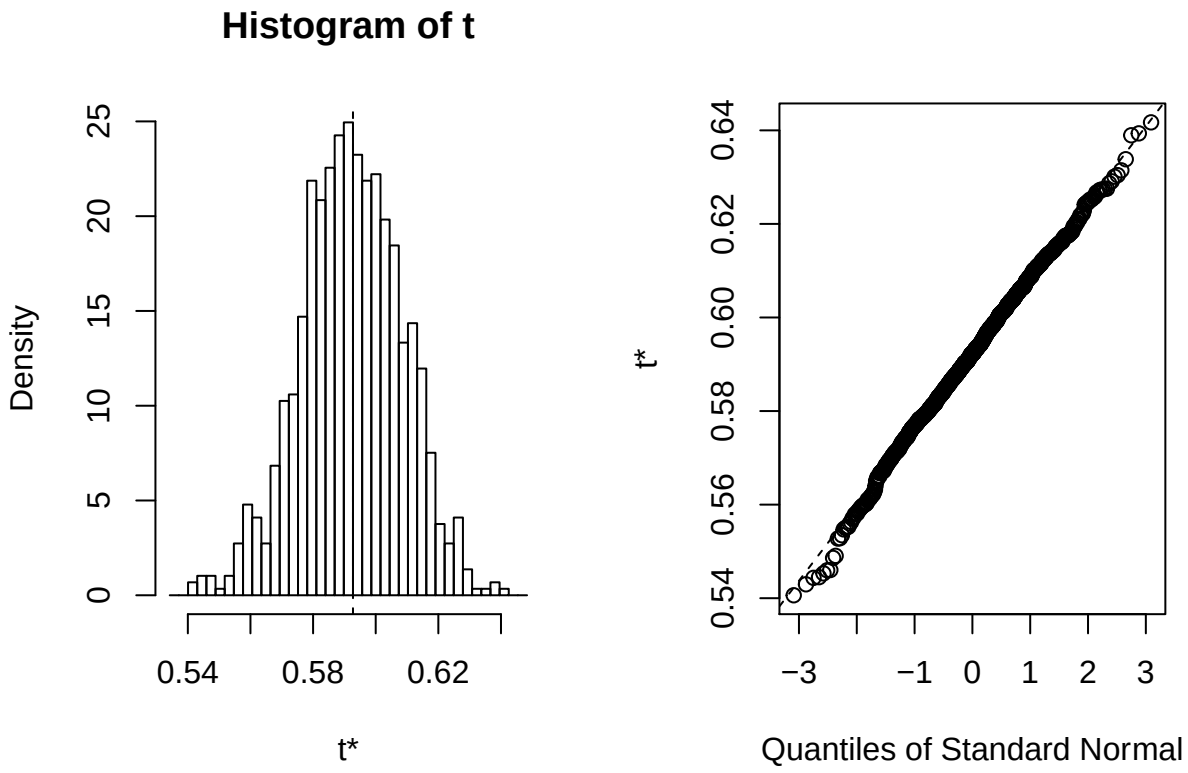
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.5656,  0.6309 )
## Calculations and Intervals on Original Scale
```

3.6 A/B Together, Coded Norminally

```
b <- boot(data = AB.together, statistic = nominal.alpha, R = 1000)
bs
```

```
## Error in eval(expr, envir, enclos): object 'bs' not found
```

```
plot(b)
```




```
boot.ci(b, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = b, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.5589,  0.6243 )
## Calculations and Intervals on Original Scale
```