Krippendorff's alpha

Cooper Stansbury

12 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")
```

1.2 Note on Confidence Intervals and Contrast 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.

1.3 Loading/Coding Annotations

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

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

```
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 1: 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 2: 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

2 Cohen's Kappas (pairwise)

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

For each pair of annotators we calculate Cohen's Kappa (weighted and unweighted). Note that the weight matrix used is as follows:

Table 3: Weights Used for Cohen's Kappa

	A	B	Blank
A	1.00	0.75	0.00
B	0.75	1.00	0.75
Blank	0.00	0.75	1.00

```r
liz.kath <- cohen.kappa(AB.separate[, c('LIZ', 'KATHLEEN')])
knitr::kable(liz.kath$confid,
             caption="Liz v. Kathleen Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 4: Liz v. Kathleen Cohen’s Kappa (weighted and weighted)

	lower	estimate	upper
unweighted kappa	0.5212	0.5606	0.6001
weighted kappa	0.6216	0.6216	0.6216

```r
liz.kay <- cohen.kappa(AB.separate[, c('LIZ', 'KAYCEE')])
knitr::kable(liz.kay$confid,
             caption="Liz v. Kaycee Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 5: Liz v. Kaycee Cohen’s Kappa (weighted and weighted)

	lower	estimate	upper
unweighted kappa	0.5787	0.6165	0.6544
weighted kappa	0.6688	0.6688	0.6688

```r
kath.kay <- cohen.kappa(AB.separate[, c('KATHLEEN', 'KAYCEE')])
knitr::kable(kath.kay$confid,
             caption="Kathleen v. Kaycee Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 6: Kathleen v. Kaycee Cohen’s Kappa (weighted and weighted)

	lower	estimate	upper
unweighted kappa	0.5078	0.5466	0.5854
weighted kappa	0.5761	0.5874	0.5988

2.2 Weighted and Unweighted Cohen’s for A/B Together

The weight matrix used is as follows:

Table 7: Weights Used for Cohen’s Kappa

	AB	Blank
AB	1	0
Blank	0	1

```r
liz.kath <- cohen.kappa(AB.together[, c('LIZ', 'KATHLEEN')])
knitr::kable(liz.kath$confid,
             caption="Liz v. Kathleen Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 8: Liz v. Kathleen Cohen's Kappa (weighted and weighted)

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

```r
liz.kay <- cohen.kappa(AB.together[, c('LIZ', 'KAYCEE')])
knitr::kable(liz.kay$confid,
             caption="Liz v. Kaycee Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 9: Liz v. Kaycee Cohen's Kappa (weighted and weighted)

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

```r
kath.kay <- cohen.kappa(AB.together[, c('KATHLEEN', 'KAYCEE')])
knitr::kable(kath.kay$confid,
             caption="Kathleen v. Kaycee Cohen's Kappa (weighted and weighted)",
             digits = 4)
```

Table 10: Kathleen v. Kaycee Cohen's Kappa (weighted and weighted)

	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]."

4 Krippendorff's alpha A/B Separate

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

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

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

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

4.1 Krippendorff's alpha A/B Together

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

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

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

4.2 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:

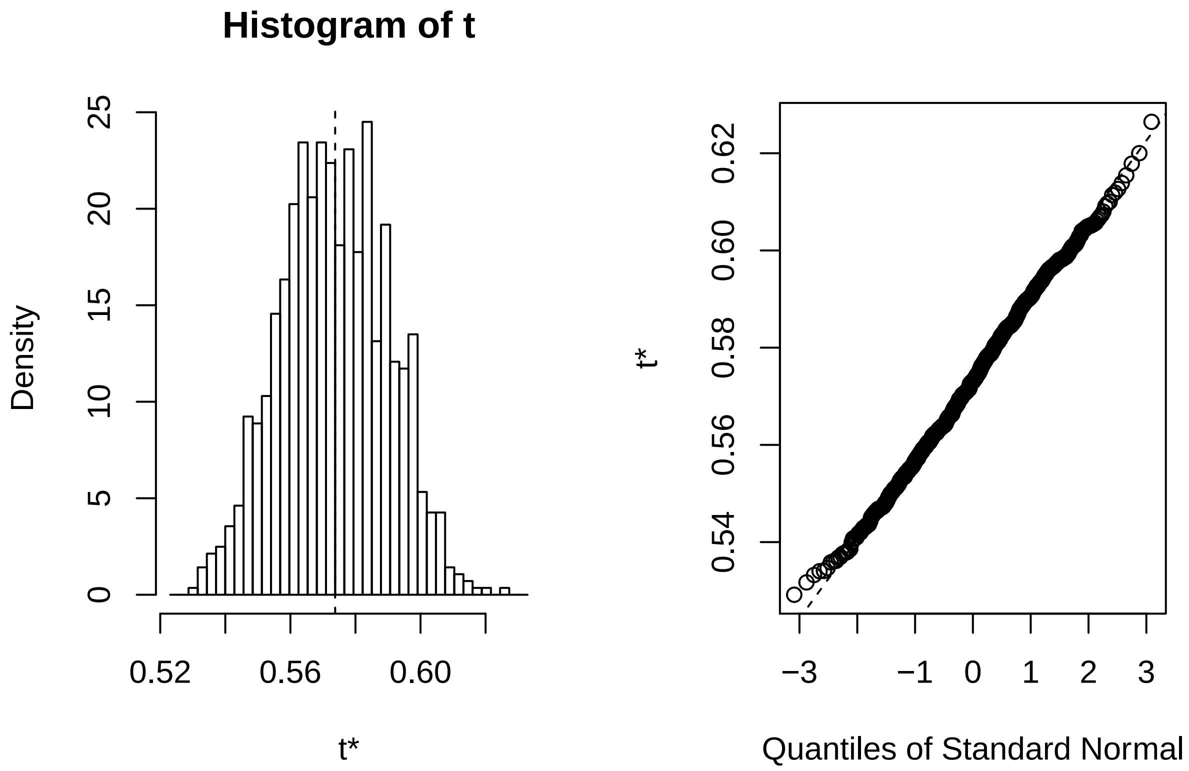
```
nominal.alpha <- function(d, w){
  #' a function bootstrap nominal coding of
  #' Krippendorff's alpha
  data <- t(d[w,])
  kripp.alpha(data, 'nominal')$value
}

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

4.2.1 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 -0.0003353678  0.01638747
```

```
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%   ( 0.5419,  0.6047 )
## Calculations and Intervals on Original Scale
```

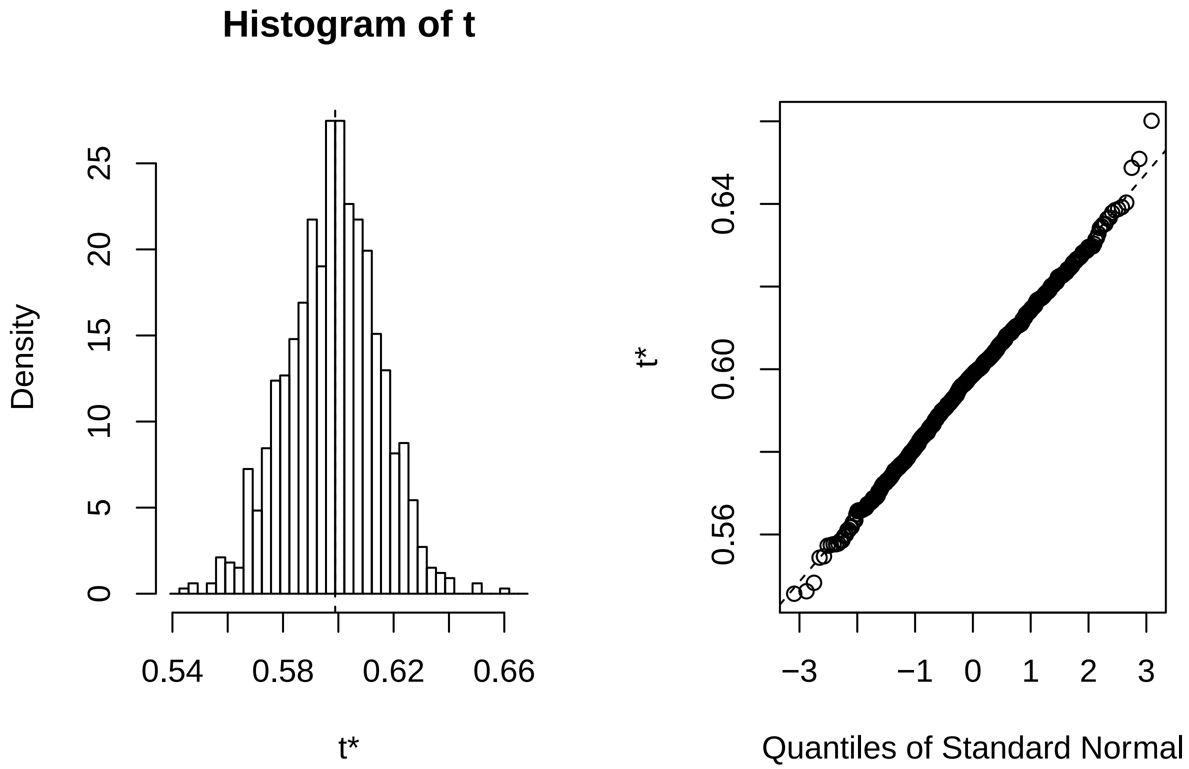
4.2.2 A/B Separate, Coded Ordinally

```
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 -0.0008249744   0.0164744
```

```
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%   ( 0.5658,  0.6286 )
## Calculations and Intervals on Original Scale
```

4.2.3 A/B Together, Coded Norminally

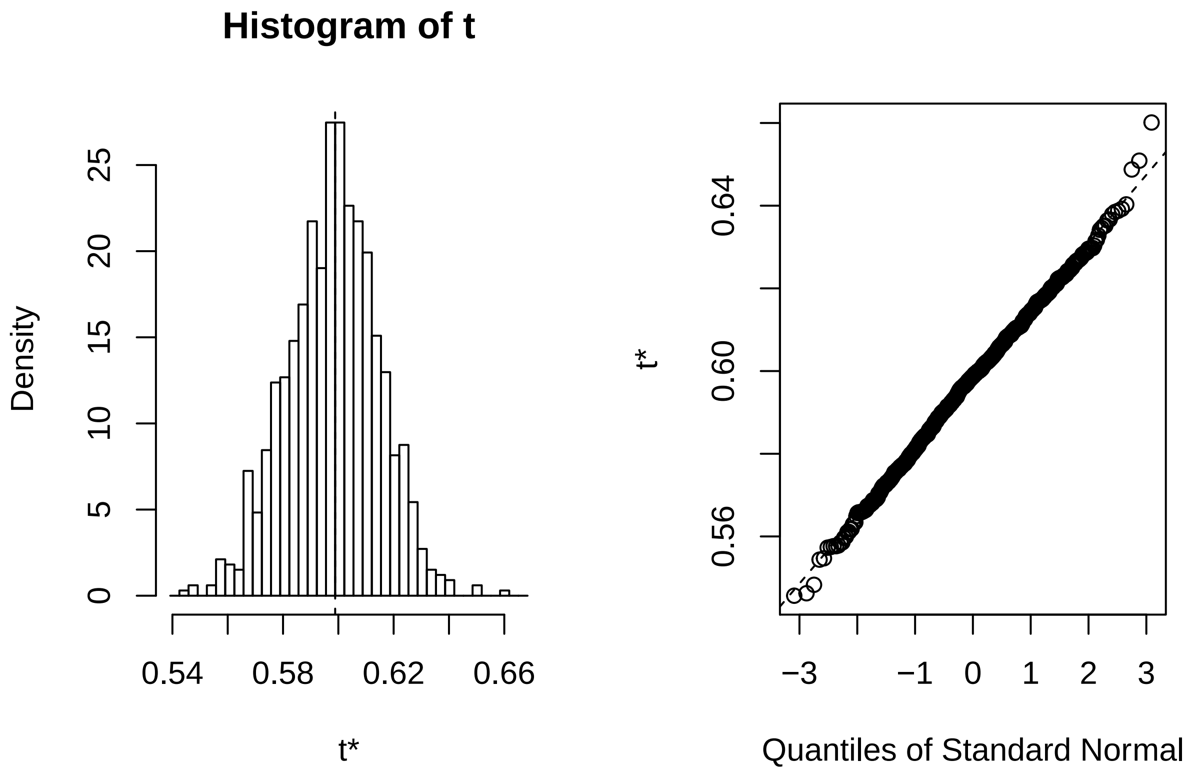
```
b <- boot(data = AAB.together, statistic = nominal.alpha, R = 1000)
```

```
## Error in NROW(data): object 'AAB.together' not found
```

```
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%   ( 0.5658,  0.6286 )
## Calculations and Intervals on Original Scale
```