**krippendorffsalpha**: An R Package for Measuring Agreement Using Krippendorff’s Alpha Coefficient

*by John Hughes*

**Abstract** R package *krippendorffsalpha* provides tools for measuring agreement using Krippendorff’s $\alpha$ coefficient, a well-known nonparametric measure of agreement (also called inter-rater reliability and various other names). This article first develops Krippendorff’s $\alpha$ in a natural way, and situates $\alpha$ among statistical procedures. Then the usage of package *krippendorffsalpha* is illustrated via analyses of two datasets, the latter of which was collected during an imaging study of hip cartilage. The package permits users to apply the $\alpha$ methodology using built-in distance functions for the nominal, ordinal, interval, or ratio levels of measurement. User-defined distance functions are also supported. The fitting function can accommodate any number of units, any number of coders, and missingness. Bootstrap inference is supported, and the bootstrap computation can be carried out in parallel.

**Introduction**

Krippendorff’s $\alpha$ (Hayes and Krippendorff, 2007) is a well-known nonparametric measure of agreement (i.e., consistency of scoring among two or more raters for the same units of analysis (Gwet, 2014)). In R (Ihaka and Gentleman, 1996), Krippendorff’s $\alpha$ can be applied using function *kripp.alpha* of package *irr* (Gamer et al., 2012), function *kripp.boot* of package *kripp.boot* (Proutskova and Gruszczynski, 2020), function *krippalpha* of package *icr* (Staudt and L’Ecuyer, 2020), and functions *krippen.alpha.raw* and *krippen.alpha.dist* of package *irrCAC* (Gwet, 2019). But these packages fail to provide a number of useful features. In this article we present package *krippendorffsalpha*, which improves upon the above mentioned packages in (at least) the following ways. Package *krippendorffsalpha*

- offers commonly used built-in distance functions for the nominal, ordinal, interval, and ratio levels of measurement, and also supports user-defined distance functions;
- conforms to the R idiom by providing S3 methods *confint*, *influence*, *plot*, and *summary*;
- supports embarrassingly parallel bootstrap computation; and
- supports verbose communication with the user, including the display of a progress bar during the production of the bootstrap sample.

The remainder of this article is organized as follows. In Section 2.2 we locate Krippendorff’s $\alpha$ among statistical procedures. In Section 2.2.1 we first develop a special case of Krippendorff’s $\alpha$ (call it $\alpha_{SED}$) in a well-known parametric setting, and then we present $\alpha$ in its most general (i.e., nonparametric) form. In Section 2.2.2 we show that $\alpha$ is a type of multiresponse permutation procedure. In Section 2.2.3 we generalize $\alpha_{SED}$ in a fully parametric fashion, arriving at Sklar’s $\omega$. In Section 2.3 we describe our package’s bootstrap inference for $\alpha$ and compare the performance of our procedure to that of two alternative approaches. In Section 2.4 we briefly discuss robustness and influence. In Section 2.5 we provide a thorough demonstration of *krippendorffsalpha*’s usage before concluding in Section 2.6.

**Situating Krippendorff’s Alpha among statistical procedures**

Since Krippendorff’s $\alpha$ is defined in terms of discrepancies (Krippendorff, 2013), at first glance one might conclude, erroneously, that $\alpha$ is a measure of *dis*-agreement and so answers the wrong question. In Sections 2.2.1–2.2.3 we will show, by examining Krippendorff’s $\alpha$’s place among statistical procedures, that $\alpha$ is, in fact, a sensible measure of agreement. Also, establishing a context for $\alpha$ may help practitioners make educated decisions regarding $\alpha$’s use.

The UML class diagram (Fowler et al., 2004) shown below in Figure 1 provides a conceptual roadmap for our development. Briefly, a special case of $\alpha$ (which we denote as Alpha(SED) or $\alpha_{SED}$) arises naturally in the context of the one-way mixed-effects ANOVA model. Alpha(SED) can then be generalized in a nonparametric fashion to arrive at Krippendorff’s $\alpha$ as it has been presented by Hayes and Krippendorff (see Gwet (2015) for a development of nonparametric $\alpha$ in terms of agreement rather
than discrepancies); this nonparametric form of α is a (slightly modified) multiresponse permutation procedure. Alternatively, αSED can be generalized in a parametric fashion to arrive at Sklar’s ω, a Gaussian copula-based methodology for measuring agreement.

Figure 1: A UML class diagram that shows the relationships between Krippendorff’s α and other statistical procedures.

**Parametric genesis of Krippendorff’s Alpha coefficient**

In this section we develop Krippendorff’s α (Hayes and Krippendorff, 2007) in an intuitive and bottom-up fashion. Our starting point is a fully parametric model, namely, the classic one-way mixed-effects ANOVA model (Ravishanker and Dey, 2001). To ease exposition we will consider only a balanced version of the model. We have, for \( n_u \) units and \( n_c \) coders, scores

\[
Y_{ij} = \mu + \tau_i + \epsilon_{ij}, \quad (i = 1, \ldots, n_u; \ j = 1, \ldots, n_c)
\]

where

- \( \mu \) (the population mean) is a fixed real number,
- the \( \tau_i \) are independent \( N(0, \sigma^2_{\tau}) \) random variables,
- the \( \epsilon_{ij} \) are independent \( N(0, \sigma^2_{\epsilon}) \) random variables, and
- the \( \tau_i \) are independent of the \( \epsilon_{ij} \).

In this setup we have \( n_c \) Gaussian codes \( Y_{i1}, \ldots, Y_{in_c} \) for unit \( i \in \{1, \ldots, n_u\} \). Conditional on \( \tau_i \), said codes are \( N(\mu + \tau_i, \sigma^2_{\epsilon}) \) random variables. Since the variables share the “unit effect” \( \tau_i \), the variables are correlated. The correlation, which is usually called the *intraclass correlation*, is given by

\[
\alpha = \frac{\sigma^2_{\tau}}{\sigma^2_{\tau} + \sigma^2_{\epsilon}} = 1 - \frac{\sigma^2_{\tau}}{\sigma^2_{\tau} + \sigma^2_{\epsilon}}.
\]

We use \( \alpha \) to denote this quantity precisely because Krippendorff’s \( \alpha \) is the intraclass correlation for codes that conform to this model. That is, for the one-way mixed-effects ANOVA model, Krippendorff’s \( \alpha \) is the intraclass correlation. The reader may recall that the estimator of \( \alpha \) for this model is

\[
\hat{\alpha} = 1 - \frac{\bar{\tau}^2}{\bar{\tau}^2 + \bar{\epsilon}^2} = 1 - \frac{1}{n_u(n_u - 1)} \sum_{i=1}^{n_u} \sum_{j=1}^{n_c} (Y_{ij} - \bar{Y}_i)^2 \sum_{i=1}^{n_u} \sum_{j=1}^{n_c} (Y_{ij} - \bar{Y}_i)^2.
\]
where $\bar{Y}_i$ and $\bar{Y}_s$ denote the arithmetic means for the $i$th unit and for the entire sample, respectively. The form of this estimator is not surprising, of course, since it is well known that assuming Gaussianity leads to variance estimators involving sums of weighted squared deviations from sample arithmetic means.

We can eliminate the arithmetic means in (1) by employing the identity

$$\sum_{i=1}^{n} \left( x_i - \bar{x} \right)^2 = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{n} \left( x_i - x_j \right)^2.$$

This gives

$$\hat{\omega} = 1 - \frac{1}{2n \rho_{\hat{\omega}}(\hat{z})} \sum_{i=1}^{n} \sum_{j=1}^{n} \left( Y_{ij} - Y_{\hat{\omega}} \right)^2.$$  \hspace{1cm} (2)

Now, let $d^2(x, y) = (x - y)^2$ and rewrite (2) as

$$\hat{\omega} = 1 - \frac{D_o}{D_e} = 1 - \frac{1}{2n \rho_{\hat{\omega}}(\hat{z})} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \sum_{i}^{n} \sum_{j}^{n} \sum_{k}^{n} \sum_{l}^{n} \left( Y_{ij} - Y_{kl} \right)^2,$$  \hspace{1cm} (3)

where $D_o$ and $D_e$ denote observed and expected disagreement, respectively. This is Krippendorff’s $\alpha$ for the squared Euclidean distance (which is not a metric but a Bregman divergence (Bregman, 1967)). (We will henceforth refer to this version of $\alpha$ as Alpha(SED) or $\alpha_{SED}$. ) As we mentioned above, this form of Krippendorff’s $\alpha$ arises quite naturally when the data at hand conform to the one-way mixed-effects ANOVA model, for which agreement corresponds to positive correlation. More generally, Krippendorff recommends this form of $\alpha$ for the interval level of measurement. For other levels of measurement, Krippendorff presents other distance functions $d^2$ (several possibilities are shown in Table 1). Note that package krippendorffalpha supports user-defined distance functions as well as the interval, nominal, and ratio distance functions shown in the table.

<table>
<thead>
<tr>
<th>Level of Measurement</th>
<th>Distance Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>$d^2(x, y) = (x - y)^2$</td>
</tr>
<tr>
<td>nominal</td>
<td>$d^2(x, y) = 1{x \neq y}$</td>
</tr>
<tr>
<td>ratio</td>
<td>$d^2(x, y) = \left( \frac{x - y}{x + y} \right)^2$</td>
</tr>
<tr>
<td>bipolar</td>
<td>$d^2(x, y) = \frac{(x - y)^2}{(x + y - 2\min)^2(2\max - x - y)}$</td>
</tr>
<tr>
<td>circular</td>
<td>$d^2(x, y) = \left( \sin \left( \frac{\pi x - y}{I} \right) \right)^2$ (I = number of equal intervals on circle)</td>
</tr>
<tr>
<td>ordinal</td>
<td>$d^2(x, y) = (x - y)^2$ (adjacent ranks are equidistant)</td>
</tr>
</tbody>
</table>

Table 1: Several distance functions that may be appropriate for use in Krippendorff’s $\alpha$.

**Alpha as a multiresponse permutation procedure**

In any case, (3) is nonparametric for arbitrary $d^2$ since then the estimator $\hat{\omega}$ does not usually correspond to a well-defined population parameter $\alpha$. This more general form of Krippendorff’s $\alpha$ is in fact a special case of the so called multiresponse permutation procedure (MRPP). The MRPPs form a class of permutation methods for discerning differences among two or more groups in one or more dimensions (Mielke and Berry, 2007). Note, however, that although $\alpha$ can be viewed as an MRPP (as we are about to show), $\alpha$ has been modified for the purpose of measuring agreement rather than discerning differences.

To show that Krippendorff’s $\alpha$ is an MRPP, we first present the general form of the MRPP. Following Mielke and Berry, let $\Omega = \{\omega_1, \ldots, \omega_n\}$ be a finite sample that is representative of some population of interest, let $S_1, \ldots, S_{n+1}$ denote a partition of $\Omega$ into $n + 1$ disjoint groups, and let $\rho$ be a metric that makes sense for the objects of $\Omega$. (Strictly speaking, $\rho$ need not be a metric; a symmetric distance function will suffice.) To ease notation a bit, let $\rho_{jk} \equiv \rho(\omega_j, \omega_k)$. Then the MRPP statistic can be written
as
\[
\delta = \sum_{i=1}^{\alpha} C_i \theta_i,
\]
where \( C_i > 0 \) are group weights such that \( \sum C_i = 1 \):
\[
\theta_i = \frac{1}{n_i} \sum_{j<k} \rho_{jk} 1_{\{\omega_j\}} 1_{\{\omega_k\}},
\]
is the average distance between distinct pairs of objects in group \( S_i \); \( n_i \geq 2 \) is the number of objects in group \( i \); \( l = \sum_{i=1}^{\alpha} n_i \); \( n_{a+1} = n - l \geq 0 \) is the number of remaining unclassified objects in group \( S_{a+1} \); and \( 1_i \) is the indicator function for membership in group \( i \).

Note that this formulation is quite general since the objects in \( \Omega \) can be scalars, vectors, or more exotic objects; and we are free to choose the metric and the weights. In the case of Krippendorff’s \( \alpha \) we can produce \( \delta = D_o \) by letting \( \rho = d^2 \) for some appropriately chosen distance function \( d^2 \), and choosing weights \( C_i = 1/n_i \).

### A parametric generalization of Alpha(SED)

In the preceding sections we generalized \( \alpha_{SED} \) in a nonparametric fashion by substituting other notions of distance for the squared Euclidean distance. Now we will present a fully parametric generalization of \( \alpha_{SED} \), namely, Sklar’s \( \omega \) (Hughes, 2018).

The statistical model underpinning Sklar’s \( \omega \) is a Gaussian copula model (Xue-Kun Song, 2000). The most general form of the model is given by
\[
\begin{align*}
Z &= (Z_1, \ldots, Z_n)^T \sim N(0, \Omega) \\
U_i &= \Phi(Z_i) \sim U(0, 1) \quad (i = 1, \ldots, n) \\
Y_i &= F_i^{-1}(U_i) \sim F_i, \quad (4)
\end{align*}
\]
where \( \Omega \) is a correlation matrix, \( \Phi \) is the standard Gaussian cdf, and \( F_i \) is the cdf for the \( i \)th outcome \( Y_i \). Note that \( U = (U_1, \ldots, U_n)^T \) is a realization of the Gaussian copula, which is to say that the \( U_i \) are marginally standard uniform and exhibit the Gaussian correlation structure defined by \( \Omega \). Since \( U_i \) is standard uniform, applying the inverse probability integral transform to \( U_i \) produces outcome \( Y_i \) having the desired marginal distribution \( F_i \).

To see that the one-way mixed-effects ANOVA model (and hence \( \alpha_{SED} \)) is a special case of Sklar’s \( \omega \), let the copula correlation matrix \( \Omega \) be block diagonal, where the \( i \)th block corresponds to the \( i \)th unit \((i = 1, \ldots, n_u)\) and has a compound symmetry structure. That is,
\[
\Omega = \text{diag}(\Omega_i),
\]
where
\[
\Omega_i = \begin{pmatrix}
c_1 & c_2 & \cdots & c_{n_c} \\
c_2 & \omega & \cdots & \omega \\
\vdots & \vdots & \ddots & \vdots \\
c_{n_c} & \omega & \cdots & 1
\end{pmatrix}.
\]
Complete the specification by letting \( F_{ij} \) \((i = 1, \ldots, n_u; j = 1, \ldots, n_c)\) be the cdf for the Gaussian distribution with mean \( \mu \) and variance \( \sigma^2 \). Then \( \omega = a_i \), the intraclass correlation coefficient.

### Inference for Krippendorff’s Alpha

Mielke and Berry describe hypothesis testing for MRPPs. Specifically, they discuss three approaches: permutation, Monte Carlo resampling, and Pearson type III moment approximation; the latter of which has significant advantages. For Krippendorff’s \( \alpha \), though, we are interested not in hypothesis testing but in interval estimation. This can be done straightforwardly and efficiently using Monte Carlo resampling. Since \( D_o \) is invariant to permutation of the scores, our resampling procedure focuses on \( D_o \) only. The algorithm proceeds as follows.

1. Collect the scores in an \( n_u \times n_c \) matrix, \( A \), where each row corresponds to a unit.
For $i \in \{1, \ldots, n_b\}$, form matrix $A_i$ by sampling, with replacement, $n_u$ rows from $A$.

3. For each $A_i$, compute $D_0^{(i)}$ using the same distance function $d^2$ that was used to compute $\hat{\alpha}$.

4. For each $D_0^{(i)}$, compute $\hat{\alpha}_i = 1 - D_0^{(i)}/D_e$.

The resulting collection $\{\hat{\alpha}_1, \ldots, \hat{\alpha}_{n_b}\}$ is a bootstrap sample for $\hat{\alpha}$, sample quantiles of which are estimated confidence limits for $\alpha$.

We carried out a number of realistic simulation experiments and found that this approach to interval estimation performs well in a wide variety of circumstances. When the true distribution of $\hat{\alpha}$ is (at least approximately) symmetric, Gwet’s closed-form expression for $\hat{V}(\hat{\alpha})$, which is implemented (for categorical data only) in package irrCAC, also performs well. By contrast, we found that the bootstrapping procedure recommended by Krippendorff (2016), which is implemented in packages kripp.boot and icr, generally performs rather poorly, producing intervals that are far too narrow (e.g., 95% intervals achieve 74% coverage).

**Robustness and interpretation**

For some levels of measurement, one may, in the interest of robustness, be tempted to replace squares with absolute values (in the distance function $d^2$). This would be advantageous if one aimed to do hypothesis testing. But for Krippendorff’s $\alpha$, using absolute values instead of squares proves disastrous, for the resulting estimator $\hat{\alpha}$ is substantially negatively biased and tends to lead to erroneous inference regarding agreement. All is not lost, however, since package krippendorffsalpha provides a means of investigating the influence on $\hat{\alpha}$ of any unit or coder (see the next section for examples).

**Illustrations**

Here we illustrate the use of krippendorffsalpha by applying Krippendorff’s $\alpha$ to a couple of datasets. We will interpret the results according to the ranges given in Table 2, but we suggest—as do Krippendorff and others (Artstein and Poesio, 2008; Landis and Koch, 1977)—that an appropriate reliability threshold may be context dependent.

<table>
<thead>
<tr>
<th>Range of Agreement</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha \leq 0.2$</td>
<td>Slight Agreement</td>
</tr>
<tr>
<td>$0.2 &lt; \alpha \leq 0.4$</td>
<td>Fair Agreement</td>
</tr>
<tr>
<td>$0.4 &lt; \alpha \leq 0.6$</td>
<td>Moderate Agreement</td>
</tr>
<tr>
<td>$0.6 &lt; \alpha \leq 0.8$</td>
<td>Substantial Agreement</td>
</tr>
<tr>
<td>$\alpha &gt; 0.8$</td>
<td>Near-Perfect Agreement</td>
</tr>
</tbody>
</table>

Table 2: Guidelines for interpreting values of an agreement coefficient.

**Nominal data analyzed previously by Krippendorff**

Consider the following data, which appear in (Krippendorff, 2013). These are nominal values (in $\{1, \ldots, 5\}$) for twelve units and four coders. The dots represent missing values.

Note that the scores for all units except the sixth are constant or nearly so. This suggests near-perfect agreement, and so we should expect $\hat{\alpha}$ to be greater than 0.8.

To apply Krippendorff’s $\alpha$ to these data, first we load package krippendorffsalpha.

```r
R> library(krippendorffsalpha)
```

krippendorffsalpha: Measuring Agreement Using Krippendorff’s Alpha Coefficient


copyright (c) 2020-2021, John Hughes
For citation information, type citation("krippendorffsalpha").
Type help(package = krippendorffsalpha) to get started.
Now we create the dataset as a matrix such that each row corresponds to a unit and each column corresponds to a coder.

R> nominal = matrix(c(1,2,3,3,2,1,4,1,2,NA,NA,NA,
+ 1,2,3,3,2,2,4,1,2,5,3,
+ NA,3,3,3,2,3,4,2,2,5,1,NA,
+ 1,2,3,3,2,4,4,1,2,5,1,NA), 12, 4)
R> nominal

[1,] 1 1 NA 1
[2,] 2 2 3 2
[3,] 3 3 3 3
[4,] 3 3 3 3
[5,] 2 2 2 2
[6,] 1 2 3 4
[7,] 4 4 4 4
[8,] 1 1 2 1
[9,] 2 2 2 2
[10,] NA 5 5 5
[11,] NA NA 1 1
[12,] NA 3 NA NA

Next we apply Krippendorff’s $\alpha$ for the nominal level of measurement. If argument level is set to "nominal", the discrete metric $d_2(x,y) = 1\{x \neq y\}$ is used by default. We do a bootstrap with sample size $n_b = 1,000$ (argument confint defaults to TRUE, and control parameter bootit defaults to 1,000). We set control parameter parallel equal to FALSE because the dataset is too small to warrant parallelization of the bootstrap computation. Finally, we set argument verbose equal to TRUE so that a progress bar is shown during the bootstrap computation. The computation took less than one second.

R> set.seed(42)
R> fit.full = krippendorffs.alpha(nominal, level = "nominal", control = list(parallel = FALSE),
+ verbose = TRUE)

|++++++++++++++++++++++++++++++| 100% elapsed=00s

As is customary in R, one can view a summary by passing the fit object to summary.krippendorffsalpha, an S3 method. If krippendorffs.alpha was called with confint = TRUE, summary displays a 95% confidence interval by default. The confidence level can be specified using argument conf.level. In any case, the quantile method (Davison and Hinkley, 1997) is used to estimate the confidence limits. Any arguments passed to summary.krippendorffsalpha via ... are passed on to R’s quantile function; this allows the user to control, for example, how the sample quantiles are computed.

R> summary(fit.full)

Krippendorff’s Alpha

Data: 12 units x 4 coders

Call:
krippendorffs.alpha(data = nominal, level = "nominal", verbose = TRUE, 
control = list(parallel = FALSE))

Control parameters:
parallel FALSE
bootit 1000

Results:

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>0.7429</td>
<td>0.4644</td>
<td>1</td>
</tr>
</tbody>
</table>

We see that \( \hat{\alpha} = 0.74 \) and \( \alpha \in (0.46, 1.00) \). This point estimate indicates only substantial agreement, which is not what we expected. At least the interval is consistent with near-perfect agreement, but we should not take this interval too seriously since the interval is rather wide (owing to the small size of the dataset).

Perhaps the substantial disagreement for the sixth unit was influential enough to yield \( \hat{\alpha} \leq 0.8 \). We can use influence.krippendorffsalpha, another S3 method, to investigate. This function, like other R versions of influence (e.g., influence.lm, influence.glm), computes DFBETA statistics (Young, 2017), as illustrated below.

R> (inf.6 = influence(fit.full, units = 6))

$dfbeta.units
  6 -0.1141961

Leaving out the sixth unit yields a DFBETA statistic of -0.11, which implies that \( \hat{\alpha} \) would have been 0.86. This is consistent with our initial hypothesis.

R> fit.full$alpha.hat - inf.6$dfbeta.units

  alpha
  0.8571429

Let us call krippendorffs.alpha again to get a new interval.

R> fit.sub = krippendorffs.alpha(nominal[-6, ], level = "nominal", 
control = list(parallel = FALSE))

confint(fit.sub)

<table>
<thead>
<tr>
<th></th>
<th>0.025</th>
<th>0.975</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6616541</td>
<td>1.0000000</td>
<td></td>
</tr>
</tbody>
</table>

We see that excluding the sixth unit leads to \( \alpha \in (0.66, 1.00) \). The new 95% interval was returned by S3 method confint.krippendorffsalpha, whose level argument defaults to 0.95, in keeping with R’s other confint methods. Note that confint.krippendorffsalpha, like summary.krippendorffsalpha, passes any ... arguments on to the quantile function.

We conclude this example by producing a visual display of our results (Figure 3). (The figure was produced via a call to S3 method plot.krippendorffsalpha, which in turn calls hist and abline, and does not show a kernel density estimate. Function plot.krippendorffsalpha is capable of producing highly customized plots; see the package documentation for details.) Since \( \hat{\alpha} \) is close to 1 and the dataset is small, the bootstrap distribution is substantially skewed to the left. Thus these data provide a textbook example of the importance of bootstrapping.

R> plot(fit.sub, xlim = c(0, 1), xlab = "Bootstrap Estimates", main = "Nominal Data", 
+ density = FALSE)

Since the dataset used in this example has missing values, we take this opportunity to explain how the package handles missingness. First, the scores for a given unit of analysis are included in the computation only if two or more scores are present for that unit. Otherwise the unit’s row of the data matrix is simply ignored. Second, if two or more scores are present for a given unit, each NA for that unit is ignored in the computations for that row. This is handled both by the loop (adjusted denominator) and by the distance function, which should return 0 if either of its arguments is NA. In the next example we illustrate this by way of a user-defined distance function, and of course the package’s built-in distance functions take the same approach.
Figure 3: A plot of the results from our analysis of the nominal data. The histogram shows the bootstrap sample, the solid orange line marks the value of \( \hat{\alpha} \), and the dashed blue lines mark the 95% confidence limits.

Interval data from an imaging study of hip cartilage

The data for this example, some of which appear in Figure 4, are 323 pairs of T2* relaxation times (a magnetic resonance quantity) for femoral cartilage (Nissi et al., 2015) in patients with femoroacetabular impingement (Figure 5), a hip condition that can lead to osteoarthritis. One measurement was taken when a contrast agent was present in the tissue, and the other measurement was taken in the absence of the agent. The aim of the study was to determine whether raw and contrast-enhanced T2* measurements agree closely enough to be interchangeable for the purpose of quantitatively assessing cartilage health.

\[
\begin{align*}
  & u_1 & u_2 & u_3 & u_4 & u_5 & \ldots & u_{319} & u_{320} & u_{321} & u_{322} & u_{323} \\
  c_1 & 27.3 & 28.5 & 29.1 & 31.2 & 33.0 & \ldots & 19.7 & 21.9 & 17.7 & 22.0 & 19.5 \\
  c_2 & 27.8 & 25.9 & 19.5 & 27.8 & 26.6 & \ldots & 18.3 & 23.1 & 18.0 & 25.7 & 21.7 
\end{align*}
\]

Figure 4: Raw and contrast-enhanced T2* values for femoral cartilage.

First we load the cartilage data, which are included in the package. The cartilage data are stored in a data frame; we convert the data frame to a matrix, which is the format required by krippendorffs.alpha.

```r
R> data(cartilage)
R> cartilage = as.matrix(cartilage)
```

Now we compute \( \hat{\alpha} \) for the interval level of measurement, i.e., squared Euclidean distance. We also produce a bootstrap sample of size 10,000. Since this dataset is much larger than the dataset analyzed in the preceding section, we parallelize the bootstrap computation. We use three CPU cores (of the four available on the author’s computer). Setting argument verbose to TRUE causes the fitting function to display a progress bar once again. The computation took five seconds to complete.
Figure 5: An illustration of femoroacetabular impingement (FAI). Top left: normal hip joint. Top right: cam type FAI (deformed femoral head). Bottom left: pincer type FAI (deformed acetabulum). Bottom right: mixed type (both deformities present).

```r
R> set.seed(12)
R> fit.sed = krippendorffs.alpha(cartilage, level = "interval", verbose = TRUE,
  + control = list(bootit = 10000, parallel = TRUE,
  + nodes = 3))
Control parameter 'type' must be "SOCK", "PVM", "MPI", or "NWS". Setting it to "SOCK".
|++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=05s

A call of function summary.krippendorffsalpha produced the output shown below.

R> summary(fit.sed)
Krippendorff's Alpha
Data: 323 units x 2 coders
Call:
krippendorffs.alpha(data = cartilage, level = "interval", verbose = TRUE,
  control = list(bootit = 10000, parallel = TRUE, nodes = 3))
Control parameters:
  bootit 10000
  parallel TRUE
  nodes 3
  type SOCK

Results:

  Estimate   Lower   Upper
alpha  0.8369  0.8080  0.8648

We see that $\hat{\alpha} = 0.84$ and $\alpha \in (0.81, 0.86)$. Thus these data suggest that raw T2* measurements agree almost perfectly with contrast-enhanced T2* measurements, perhaps rendering gadolinium-based contrast agents (GBCAs) unnecessary in T2*-based cartilage assessment. This finding could
have clinical significance since the use of GBCAs is not free of risk to patients, especially pregnant women and patients with impaired kidney function. For much additional information regarding the potential risks associated with the use of GBCAs, we refer the interested reader to the University of California, San Francisco’s policy on MRI with contrast: https://radiology.ucsf.edu/patient-care/patient-safety/contrast/mri-with-contrast-gadolinium-policy.

Figure 6 provides a visual display of the cartilage results. The histogram and kernel density estimate show the expected large-sample behavior of \( \hat{\alpha} \), i.e., the estimator is approximately Gaussian-distributed and has a small variance.

![Cartilage Data (SED)](image)

**Figure 6:** A plot of the results from our analysis of the cartilage data. The histogram and kernel density estimate (dotted black curve) show the bootstrap sample, the solid orange line marks the value of \( \hat{\alpha} \), and the dashed blue lines mark the 95% confidence limits.

We mentioned above that attempting to robustify Krippendorff’s \( \alpha \) by using absolute values in place of squares may prove problematic. This is evident for the cartilage data, as we now demonstrate.

First, define a new distance function as follows. Note that any user-defined distance function must deal explicitly with \( \text{NA} \)s if the data at hand exhibit missingness. There are no missing values in the cartilage data, but we illustrate the handling of \( \text{NA} \) anyway.

```r
R> L1.dist = function(x, y)
+ {
+   d = abs(x - y)
+   if (is.na(d))
+     d = 0
+   d
+ }
```

Now we call `krippendorffs.alpha`, supplying our new distance function via the `level` argument.

```r
R> fit.L1 = krippendorffs.alpha(cartilage, level = L1.dist, verbose = TRUE,
+                               control = list(bootit = 10000, parallel = TRUE,
+                                            nodes = 3))
```

Control parameter 'type' must be "SOCK", "PVM", "MPI", or "NWS". Setting it to "SOCK".
The results are summarized below. These results strongly suggest that only moderate to substantial agreement exists between raw T2* measurements and contrast-enhanced T2* measurements. This contradicts not only our $\alpha_{\text{SED}}$ analysis but also a Sklar’s $\omega$ analysis that assumed a non-central t marginal distribution to accommodate slight asymmetry.

```
R> summary(fit.L1)
Krippendorff’s Alpha

Data: 323 units x 2 coders

Call:
krippendorffs.alpha(data = cartilage, level = L1.dist, verbose = TRUE,
                   control = list(bootit = 10000, parallel = TRUE, nodes = 3))

Control parameters:
  bootit 10000
  parallel TRUE
  nodes 3
  type SOCK

Results:

    Estimate Lower Upper
  alpha   0.6125 0.5761 0.648
```

**Summary and discussion**

In this article we described Krippendorff’s $\alpha$ methodology for measuring agreement, and illustrated the use of R package `krippendorffsalpha`. We first established $\alpha$‘s context among statistical procedures. Specifically, the one-way mixed-effects ANOVA model provides a natural, intuitive genesis for $\alpha$ as the intraclass correlation coefficient. This form of $\alpha$ can be generalized in a parametric fashion to arrive at Sklar’s $\omega$, or in a nonparametric fashion to arrive at the form of $\alpha$ presented by Krippendorff, which is a special case of the multiresponse permutation procedure.

We demonstrated the use of `krippendorffsalpha` version 1.1 by analyzing two datasets: a nominal dataset previously analyzed by Krippendorff, and a sample of raw and contrast-enhanced T2* values from an MRI study of hip cartilage. These analyses highlighted the benefits of the package, which include the use of S3 methods, parallel bootstrap computation, support for user-defined distance functions, and a means of identifying influential units and/or coders.

**Computational details**

The results in this paper were obtained using R 4.0.3 for macOS, and the `pbapply` 1.4-2 package. R itself and all packages used (save `kripp.boot`) are available from the Comprehensive R Archive Network (CRAN) at [http://CRAN.R-project.org](http://CRAN.R-project.org). Package `krippendorffsalpha` may be downloaded from CRAN or from the author’s GitHub repository, which can be found at [https://github.com/drjphughesjr/krippendorffsalpha](https://github.com/drjphughesjr/krippendorffsalpha). Information about the author’s other R packages can be found at [http://www.johnhughes.org/software.html](http://www.johnhughes.org/software.html).

*John Hughes*

*Department of Statistics*  
*The Pennsylvania State University*  
*University Park, PA*  
*USA*  
*drjphughesjr@gmail.com*
Bibliography


