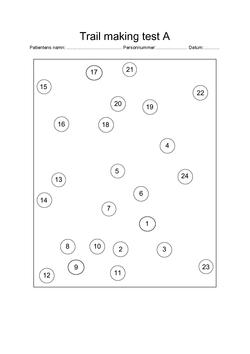
Psychometrics: Validity and Reliability

Shane T. Mueller [shanem@mtu.edu](mailto:shanem@mtu.edu)

2019-02-05

## Measures of Reliability and Validity

Reliability and validity go hand-in-hand, but they are somewhat independent. ‘’Reliability’’ refers to the extent to which your measure is reproducible, and likely to give you the same answer if you applied it twice (assuming you were able to apply it twice). ‘’Validity’’ refers to the extent to which the measure suits the purpose you are using it for. Many forms of both reliability and validity have been described, and although validity is typically a more conceptual property, both properties are established by looking at the psychometric properties of a measure. We will look at several ways of assessing the psychometric properties of a test. First, we will look at repeated measurement of the “trail-making” test.



The trailmaking test. Patients play connect-the-dots, either with a series of numbers (Form A) or a mixed series of numbers and letters (Form B).

## Test-retest, inter-rater, and inter-test reliability

If you want to know whether a test will give the same results if you measure it more than once, the simplest way to do this is to measure it twice and look at the correlation between the measures. This is known as test-retest reliability. Often, people will use standard correlation coefficients.

This can come in handy in several types of situations, including:

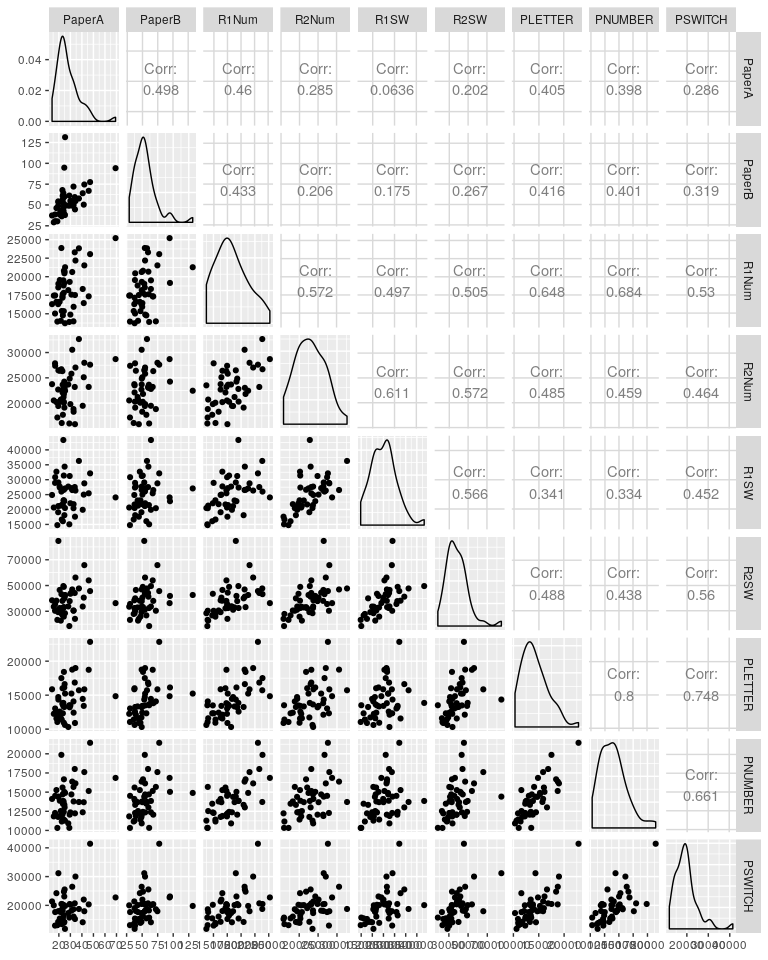
* When each person has taken the same test twice (maybe with delay)
* When you have developed different forms of a test and want to assess whether they measure the same thing.
* When you have two raters coding behavior and want to establish inter-rater reliability
* When you want to establish criterion validity of one measure against a second normed measure
* When you want to determine whether what you are measuring in the first five minutes of a test is the same as what an entire 30-minute test measures.
* Comparing one measure of a construct to another measure of a construct (e.g., executive function; spatial reasoning, etc.).
* Compare a computerized test to a paper-and-pencil test

## Standard correlation coefficients

The simplest approach is to compute the correlation between your two variables. If your variables are significantly non-normal (e.g., highly-skewed), you might use a spearman (rank-order) correlation instead of a continuous Pearson correlation. If your two measures are likely to be on different scales (e.g., if you are comparing one existing measure of vision to a new measure of vision), this is also appropriate, but recognize that a high correlation does not mean the two measures have the same values.

Consider the following data set using several related measures of the trail-making test (TMT). In this study, participants completed 9 TMT measures. The TMT involves doing a connect-the-dots test, and compares one (Form A) that is just letters A-B-C-D, and a second (Form B) that rotates between letters and number (A-1-B-2-…). First, they completed the standard Reitan paper-and-pencil test, initially Form A and then Form B. However, along with having a different layout, these forms are also of different lengths and complexities, which is a problem for the test—but one most people ignore (even though it is one of the most widely used cognitive tests in existence). To help develop a better alternative, we then had participants solve four versions of the test via computer–two each for both layouts in both letter conditions. Finally, several additional new randomly-generated tests were completed to provide independent pure and switch scores. We are interested in whether the different versions in either a switch or a non-switch condition have high test-retest validity–do they measure similar things?

library(ggplot2)  
library(GGally) ##GGally is a front-end to ggplot that will make the pairs plot equivalent  
library(reshape2)  
tmt <- read.csv("trailmaking.csv")  
#pairs(tmt[,-1])  
ggpairs(tmt[,-1])



library(knitr)  
kable(round(cor(tmt[,-1]),2),caption="Pearson Inter-correlations between different trail-making tests")

Pearson Inter-correlations between different trail-making tests

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PaperA | PaperB | R1Num | R2Num | R1SW | R2SW | PLETTER | PNUMBER | PSWITCH |
| PaperA | 1.00 | 0.50 | 0.46 | 0.29 | 0.06 | 0.20 | 0.40 | 0.40 | 0.29 |
| PaperB | 0.50 | 1.00 | 0.43 | 0.21 | 0.18 | 0.27 | 0.42 | 0.40 | 0.32 |
| R1Num | 0.46 | 0.43 | 1.00 | 0.57 | 0.50 | 0.50 | 0.65 | 0.68 | 0.53 |
| R2Num | 0.29 | 0.21 | 0.57 | 1.00 | 0.61 | 0.57 | 0.48 | 0.46 | 0.46 |
| R1SW | 0.06 | 0.18 | 0.50 | 0.61 | 1.00 | 0.57 | 0.34 | 0.33 | 0.45 |
| R2SW | 0.20 | 0.27 | 0.50 | 0.57 | 0.57 | 1.00 | 0.49 | 0.44 | 0.56 |
| PLETTER | 0.40 | 0.42 | 0.65 | 0.48 | 0.34 | 0.49 | 1.00 | 0.80 | 0.75 |
| PNUMBER | 0.40 | 0.40 | 0.68 | 0.46 | 0.33 | 0.44 | 0.80 | 1.00 | 0.66 |
| PSWITCH | 0.29 | 0.32 | 0.53 | 0.46 | 0.45 | 0.56 | 0.75 | 0.66 | 1.00 |

The Spearman rank-order correlation might be better because these are times, which are likely to be skewed positive:

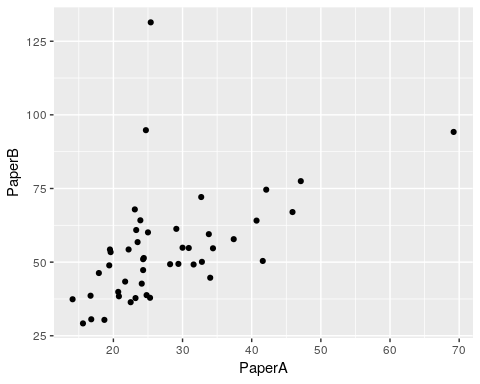
kable(round(cor(tmt[,-1],method="spearman"),2),caption="Spearman rank-order correlations")

Spearman rank-order correlations

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PaperA | PaperB | R1Num | R2Num | R1SW | R2SW | PLETTER | PNUMBER | PSWITCH |
| PaperA | 1.00 | 0.61 | 0.36 | 0.16 | 0.07 | 0.32 | 0.40 | 0.30 | 0.22 |
| PaperB | 0.61 | 1.00 | 0.41 | 0.21 | 0.20 | 0.41 | 0.59 | 0.45 | 0.41 |
| R1Num | 0.36 | 0.41 | 1.00 | 0.51 | 0.53 | 0.63 | 0.64 | 0.61 | 0.54 |
| R2Num | 0.16 | 0.21 | 0.51 | 1.00 | 0.64 | 0.62 | 0.46 | 0.42 | 0.50 |
| R1SW | 0.07 | 0.20 | 0.53 | 0.64 | 1.00 | 0.69 | 0.33 | 0.35 | 0.49 |
| R2SW | 0.32 | 0.41 | 0.63 | 0.62 | 0.69 | 1.00 | 0.63 | 0.51 | 0.64 |
| PLETTER | 0.40 | 0.59 | 0.64 | 0.46 | 0.33 | 0.63 | 1.00 | 0.77 | 0.70 |
| PNUMBER | 0.30 | 0.45 | 0.61 | 0.42 | 0.35 | 0.51 | 0.77 | 1.00 | 0.64 |
| PSWITCH | 0.22 | 0.41 | 0.54 | 0.50 | 0.49 | 0.64 | 0.70 | 0.64 | 1.00 |

Here, the rank-order Spearman correlation coefficient is a little higher (on average, about .027), but a lot higher for the Paper A versus B (.61 versus .49). We can use these correlations to examine the extent to which we have high positive correlations between forms.

ggplot(tmt,aes(x=PaperA,y=PaperB))+geom\_point()



Results show:

* Correlation of .61 between form A and B paper tests
* Low correlation between paper and computer versions of the same problem. Here, form A paper is identical to R1Num (), and Form B paper is identical to R2SW ().
* The only correlations that start getting interesting are between computerized tests.
* Two forms of the simple PEBL Test were letter and number. Their correlation was +.8, indicating fairly good alternative-form reliability. The correlated moderately strongly with other computerized measures, and less well with the paper-and-pencil tests.
* Tests requiring a switch do not stand out as especially related to one another.

# Intra-class Correlation Coefficients

When measures are on the same scale, you should use intraclass correlation coefficient (ICC). In addition, ICC will compute a value when you have two ‘or more’ measures of interest.

To understand how ICC works, you have to remember back to one-way ANOVA, where we computed of the overall model.

In this data set, do we have things on the same scale? In a sense we do–they are not all bound on a likert scale, but there was always a significant different between methods, and often a difference between forms/instructions within one method. But since we have three different measure–the paper-and-pencil tests; computerized versions of comparable tests, we can compute ICC and see what happens:

The ICC package computes ICC, but requires a long data format. This data is a bit atypical–we would usually want to use ICC when we have two measures of the same thing from each person; we have two different measures that we are hoping are the same but we know are really probably different.

require(ICC)  
library(reshape2)  
library(dplyr)  
library(ICC)  
  
##DO NOT DO THIS--it is wrong:  
#ICCest(PaperA, PaperB,tmt)  
#ICCest(R1Num, R2Num,tmt)  
  
head(melt(dplyr::select(tmt,Subject,PaperA, PaperB),"Subject"))

Subject variable value  
1 3026 PaperA 23.3  
2 3027 PaperA 33.8  
3 3028 PaperA 25.3  
4 3029 PaperA 32.8  
5 3030 PaperA 18.7  
6 3031 PaperA 37.4

tmt$Subject <- as.factor(tmt$Subject)  
##ICC on subject group:  
ICCest(x=Subject,y=value,data=melt(dplyr::select(tmt,Subject,PaperA, PaperB),"Subject"))

$ICC  
[1] -0.2092908  
  
$LowerCI  
[1] -0.466982  
  
$UpperCI  
[1] 0.08212014  
  
$N  
[1] 46  
  
$k  
[1] 2  
  
$varw  
[1] 487.7401  
  
$vara  
[1] -84.41272

cor(tmt$PaperA,tmt$PaperB) #compare ICC to this.

[1] 0.4979085

In the above, we see a negative ICC value, which does not make a lot of sense. The traditional formula for ICC is a ratio of variances: the variance related to the subject, divided by the sum of the variances related to subject, repeated treatment, and error. According to that formula, ICC cannot be below 0.

However, as explained by the following, ICC is usually estimated with sample mean squares:

<https://stats.stackexchange.com/questions/214124/what-to-do-with-negative-icc-values-adjust-the-test-or-interpret-it-differently>

The ICC formula becomes:

where is subject-related, is treatment-related, and is error-related

Here, if is smaller than , and is small,the denominator can become negative. A negative ICC essentially indicates an ICC of 0, although other measures have been proposed that avoid this issue (see James, L. R., Demaree, R. G., & Wolf, G. (1984). Estimating within-group interrater reliability with and without response bias. Journal of Applied Psychology, 69(1), 85–98.). The RWG measure is availaable as part of the `multilevel’ package, along with two measures of ICC (ICC1: individual variance explained by group membership; and ICC2: reliability of group means)

library(multilevel)  
dat <-melt(dplyr::select(tmt,Subject,PaperA, PaperB),"Subject")  
model <- aov(value~Subject,data=dat)  
cat("ICC1 from multilevel:\n")

ICC1 from multilevel:

ICC1(model)

[1] -0.2092908

cat("ICC2 from multilevel:\n")

ICC2 from multilevel:

ICC2(model)

[1] -0.529375

##consult documentation for evaluating this:  
summary(rwg(dat$value,dat$Subject))

grpid rwg gsize   
 3026 : 1 Min. :0 Min. :2   
 3027 : 1 1st Qu.:0 1st Qu.:2   
 3028 : 1 Median :0 Median :2   
 3029 : 1 Mean :0 Mean :2   
 3030 : 1 3rd Qu.:0 3rd Qu.:2   
 3031 : 1 Max. :0 Max. :2   
 (Other):40

Be careful about the grouping variable. ICC can be used in a lot of ways, not all of them make sense. We could give it the test variable (named variable) as the grouping, but this would measure whether tests differ consistently across people, rather than people differ consistently across tests.

Returning to ICC from the ICC package, lets consider paper A versus B tests, but reversing the ICC calculation and giving it the test rather than the subject as the grouping variable:

##ICC on test group:  
ICCest(variable,value,data=melt(dplyr::select(tmt,Subject,PaperA, PaperB),"Subject"))

$ICC  
[1] 0.6119805  
  
$LowerCI  
[1] 0.2223778  
  
$UpperCI  
[1] 0.9993827  
  
$N  
[1] 2  
  
$k  
[1] 46  
  
$varw  
[1] 224.9284  
  
$vara  
[1] 354.7548

This looks good, but it is difficult to interpret.

ICC is based on the logic of regression and ANOVA. Look at the when we do the related regression:

tmp <- melt(dplyr::select(tmt,Subject,PaperA, PaperB),"Subject")  
test1 <- lm(value~as.factor(Subject),data=tmp)  
summary(test1)

Call:  
lm(formula = value ~ as.factor(Subject), data = tmp)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-53.00 -11.66 0.00 11.66 53.00   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 42.10 15.62 2.696 0.00977 \*\*  
as.factor(Subject)3027 4.55 22.08 0.206 0.83768   
as.factor(Subject)3028 -10.50 22.08 -0.475 0.63672   
as.factor(Subject)3029 -0.65 22.08 -0.029 0.97665   
as.factor(Subject)3030 -17.55 22.08 -0.795 0.43089   
as.factor(Subject)3031 5.50 22.08 0.249 0.80444   
as.factor(Subject)3032 -11.80 22.08 -0.534 0.59570   
as.factor(Subject)3033 0.75 22.08 0.034 0.97306   
as.factor(Subject)3034 -9.55 22.08 -0.432 0.66745   
as.factor(Subject)3035 -11.60 22.08 -0.525 0.60193   
as.factor(Subject)3036 -4.20 22.08 -0.190 0.85001   
as.factor(Subject)3037 -6.30 22.08 -0.285 0.77672   
as.factor(Subject)3038 -3.35 22.08 -0.152 0.88010   
as.factor(Subject)3039 3.40 22.08 0.154 0.87832   
as.factor(Subject)3040 14.35 22.08 0.650 0.51907   
 [ reached getOption("max.print") -- omitted 31 rows ]  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 22.08 on 46 degrees of freedom  
Multiple R-squared: 0.3901, Adjusted R-squared: -0.2065   
F-statistic: 0.6539 on 45 and 46 DF, p-value: 0.9218

Notice that her, multiple is .39, but the adjusted -.2065. These are not the same as the ICC, but ICC value is very similar to the adjusted . The issue is that subject, in this case, accounts for very little of the variance across measures, especially when corrected for the number of predictors:

test2 <- aov(value~as.factor(Subject)+variable,data=tmp)  
summary(test2)

Df Sum Sq Mean Sq F value Pr(>F)   
as.factor(Subject) 45 14351 319 2.436 0.00174 \*\*   
variable 1 16544 16544 126.343 1.18e-14 \*\*\*  
Residuals 45 5892 131   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

aggregate(value~variable,data=tmp,mean)

variable value  
1 PaperA 27.74783  
2 PaperB 54.56739

Here, test (variable) is a significant predictor, accounting for 16544 SS. This means that the two measures had overall mean difference. In this case, this was actually by design, testA and testB are different, and one is harder than the other (producing larger values). If we look at it, we see that the variable predictor accounts for as much variance (with one degree of freedom) as the subjects. ICC assumes repated independent stationary measurement, and if you have learning or fatigue it might be misleading:

R1num vs R2num are more equivalent–they are repetitions of a number-only version of the test, using two different forms.

ICCest(x=Subject,y=value,data=melt(dplyr::select(tmt,Subject,R1Num, R2Num),"Subject"))

$ICC  
[1] 0.0747591  
  
$LowerCI  
[1] -0.2154458  
  
$UpperCI  
[1] 0.3536715  
  
$N  
[1] 46  
  
$k  
[1] 2  
  
$varw  
[1] 16293352  
  
$vara  
[1] 1316496

cor(tmt$R1Num,tmt$R2Num) #<<--compare to:

[1] 0.572493

colMeans(tmt[,4:5])

R1Num R2Num   
18214.37 22886.00

Again, even though instructions were the same, and correlation between the two is reasonable, the ICC is low because there is a (smaller) mean difference–one form was more difficult. This would not be an appropriate use of ICC to establish reliability.

Finally, we might compare two tests that are intended to be equally difficult, but use numbers versus letters (which we hope won’t matter):

ICCest(Subject,value,data=melt(dplyr::select(tmt,Subject,PLETTER, PNUMBER),"Subject"))

$ICC  
[1] 0.7964547  
  
$LowerCI  
[1] 0.6612699  
  
$UpperCI  
[1] 0.8817495  
  
$N  
[1] 46  
  
$k  
[1] 2  
  
$varw  
[1] 1282400  
  
$vara  
[1] 5017917

cor(tmt$PLETTER,tmt$PNUMBER) #correlation

[1] 0.8002772

##Check whether the means differ:  
colMeans(dplyr::select(tmt,PLETTER,PNUMBER))

PLETTER PNUMBER   
14204.04 13980.67

t.test(tmt$PLETTER,tmt$PNUMBER,pair=T)

Paired t-test  
  
data: tmt$PLETTER and tmt$PNUMBER  
t = 0.94486, df = 45, p-value = 0.3498  
alternative hypothesis: true difference in means is not equal to 0  
95 percent confidence interval:  
 -252.7722 699.5114  
sample estimates:  
mean of the differences   
 223.3696

The mean values of these are pretty close, so ICC is useful. Now, cor is pretty close to ICC, and is reasonably high. Consequently, using ICC is justifiable. In the other two examples, given the difference in timing of PaperA/PaperB, ICC is probably not appropriate; the same goes for R1Num/R2Num.

##Three or more measurements Suppose we had a third observation that was related to the others:

tmt$PNUMBER2 <- (tmt$PNUMBER + tmt$PLETTER)/2  
data <- melt(dplyr::select(tmt,Subject,PNUMBER,PLETTER,PNUMBER2),"Subject")  
ICCest(Subject,value,data=data)

$ICC  
[1] 0.8946536  
  
$LowerCI  
[1] 0.8357981  
  
$UpperCI  
[1] 0.936263  
  
$N  
[1] 46  
  
$k  
[1] 3  
  
$varw  
[1] 641199.8  
  
$vara  
[1] 5445384

This is the other advantage of ICC–it will give you an overall correspondence value when you have multiple measures. Note that ICC attempts to estimate the consistency of a single measurement event, based on multiple measures. Sometimes, you instead want to measure the consistency of a bunch of measures–maybe a set of questions in a personality questionnaire. Here, the reliability of a single measure is less interesting, and instead you want to measure the reliability of the whole set. The technique for doing this (which is actually very closely related to ICC) is referred to as ‘internal consistency’, and the most common measure of it is a statistic called Cronbach’s . In fact, in some stats programs such as SPSS, ICC routines will give you a measure referred to as the “Average Measure”, in addition to the “Individual Measure”. The individual measure is what we have calculated here–estimated reliability if you are measuring the value once. The Average measure is the estimated reliability if, in the future, you really intend to measure it as many times as you did. This average measure is identical to Cronbach’s .

# Measuring Internal Consistency

When your measurement method has –a single set of materials, test-retest validity is difficult to establish because you might get carryover effects. People can learn the materials and be impacted later. Or, for personality-type questionnaires, people may remember their answers and be more consistent than they really should be. Nevertheless, for a scale–a set of questions intended to all measure a coherent construct, we’d like to get a measure of how well they go together, and indirectly a measure of reliability.

There are a number of measures of so-called “Internal consistency”, the most popular among these is Cronbach’s . These are sometimes discussed as measuring the average of all possible split-half correlations, but that definition is confusing, because it is not clear whether you are splitting questions and comparing it over people, or splitting people and comparing it over questions. The coefficient can be thought of as a measure you would get by dividing your questions into two groups and computing a correlation between them, then repeating this for all possible splits of two groups, adjusted for the total number of items you are measuring. Alternately, as described earlier, it is an adjustment to ICC estimating ICC for the entire set of questions instead of just one.

It is important to recognize that the statistics of consistency make the assumption that there is a single factor underlying the measure, and produce a statistic based on this assumption. Thus, they do not provide strong evidence in favor of a single factor structure when the values are high, but rather measure the consistency based on the assumption that there is a single factor. There are thus ways the statistics can be ‘fooled’. For example, goes up as the number of items on the scale go up. In fact, when argueing for researchers to use the Greatest Lower Bound statistic, Sijtsma (2009) argues: ``The only reason to report alpha is that top journals tend to accept articles that use statistical methods that have been around for a long time such as alpha. Reporting alpha in addition to a greater lower bound may be a good strategy to introduce and promote a better reliability estimation practice.’’

To examine some alternatives, let’s load a data set using an intelligence/reasoning task similar to Raven’s progressive matrices. These are thought to be a good measure of fluid intelligence. In this data set, we can’t take the results too seriously, because we have fewer people recorded than items on the test. In general, a serious analysis like this should have several hundred observations for a set of 20-50 questions.

library(psych)  
library(dplyr)  
library(reshape2)  
dat.raw <- read.csv("reasoning-dat.csv",  
 header=F)  
  
dat.long <- transmute(dat.raw,sub=V1,prob=V13,correct=V24)  
dat <- dcast(dat.long,sub~prob,value.var="correct")  
items <- dat[,-1]

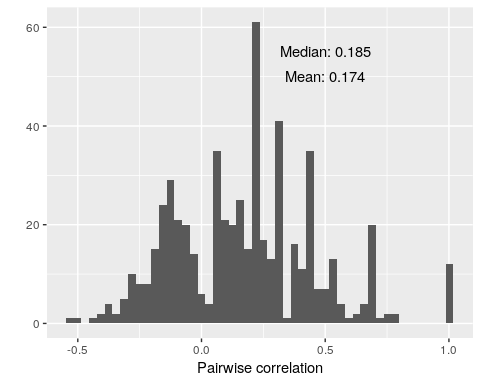
Several of the items were correct by everybody, which can cause trouble. These are not predictive of anything. So let’s get rid of them:

items2 <- items[,colMeans(items)<1]

Notice that the measures here are all 0 or 1–whether they got the score correct or not. This will necessarily limit the predictability of any single question.

#Mean Correlation One value to consider is the mean or median inter-item correlation.

library(ggplot2)  
item.cor <- cor(items2,use="pairwise.complete")  
cor.values <- item.cor[upper.tri(item.cor)]  
qplot(cor.values,bins=50) +  
 annotate(geom="text", x=.5, y=50,   
 label=paste("Mean:", round(mean(cor.values),3))) +  
 annotate(geom="text", x=.5, y=55,   
 label=paste("Median:", round(median(cor.values),3))) +  
 xlab("Pairwise correlation")



mean(cor.values)

[1] 0.17364

median(cor.values)

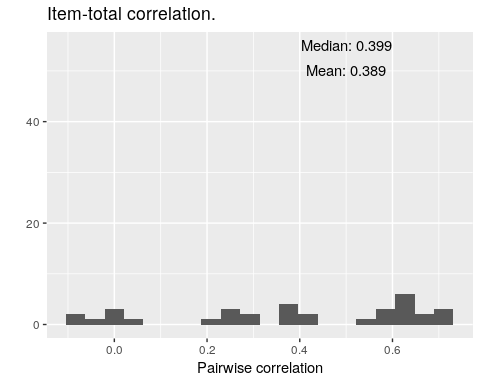
[1] 0.1846372

This shows that there is fairly weak relationships between items. Many pairs are negatively correlated, and both the mean and median correlations are around .18. On an individual basis, this is not great, but sometimes this as good as we get with single items. Also, see that there are some pairs of items that have a perfect correlation. In our case, this is most likely caused by the small number of participants, but it could also indicate redundancy and might cause problems for some of our methods. In practice, it could also mean that you have asked the same question twice, and one of them could be removed.

# Item-total correlation

Another simple measure is to compute the correlation between each item and the total score for each person. In addition to the its ease of interpretation, it can also be used to select variables for inclusion or removal.

library(ggplot2)  
library(dplyr)  
  
##Make item total, subtracting out the item score.  
itemTotal <- matrix(rep(rowSums(items2),ncol(items2)),ncol=ncol(items2)) - items2  
  
item.cor2 <- diag(cor(items2,itemTotal,use="pairwise.complete"))  
qplot(item.cor2,bins=20) +  
 annotate(geom="text", x=.5, y=50,   
 label=paste("Mean:", round(mean(item.cor2),3))) +  
 annotate(geom="text", x=.5, y=55,   
 label=paste("Median:", round(median(item.cor2),3))) +  
 xlab("Pairwise correlation") + ggtitle("Item-total correlation.")



mean(item.cor2)

[1] 0.3888878

median(item.cor2)

[1] 0.3989331

This shows how the distribution of scores on each single item correlates (across people) with the total. A high value here means that the question is very representative of the entire set. Possible, it means that you could replace the entire set with that one question, or a small number of questions that are representative of the entire set.

The results here are a bit more reassuring. On average, the correlation between any one item and the total is about 0.4. What’s more, there are a few near 0, and some that are negative, which may indicate that we have “bad” items that we shouldn’t be using. (In this case, it is more likely to stem from the fact that we have a very low number of observations.) Also, about ten questions have item-total correlations around 0.6. Maybe we could make an abbreviated test with just ten questions that is as good as the 40-item test.

# Cronbach’s

These previous measures are simple heuristics, but several measures have been developed to give a single measure about the coherence of a set of items. The most famous of these is Cronbach’s (alpha), which is appropriate for continuous (or at least ordinal)-scale measures, including response time, Likert-style questions, and the like.

Many functions related to psychometrics are available within the package. Other packages including include psy (which also has ICC and a kappa), the fmsb library, the psychometric package, and probably others.

The package has a function called alpha, which completes computes and a more complete reliability analysis, including some of the measures above.

library(psych)  
psych::alpha(items2)

Some items ( A3 B2 B5C4D3 C3 D3 D5 ) were negatively correlated with the total scale and   
probably should be reversed.   
To do this, run the function again with the 'check.keys=TRUE' option

Reliability analysis   
Call: psych::alpha(x = items2)  
  
 raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
 0.86 0.88 1 0.17 7.1 0.049 0.68 0.17 0.18  
  
 lower alpha upper 95% confidence boundaries  
0.76 0.86 0.96   
  
 Reliability if an item is dropped:  
 raw\_alpha std.alpha G6(smc) average\_r S/N var.r med.r  
A1B3 0.85 0.87 1 0.17 6.6 0.078 0.18  
A2C3 0.85 0.87 1 0.16 6.5 0.080 0.17  
A3 0.87 0.88 1 0.19 7.5 0.082 0.21  
A3B5C4 0.86 0.87 1 0.17 6.9 0.087 0.19  
A3B5E4 0.86 0.87 1 0.17 6.9 0.086 0.18  
A3C5D4 0.85 0.87 1 0.17 6.6 0.085 0.17  
A3C5E4 0.87 0.88 1 0.18 7.4 0.083 0.19  
A3D5E4 0.85 0.87 1 0.17 6.7 0.085 0.17  
A4B3D5 0.85 0.87 1 0.17 6.8 0.085 0.17  
B1E2 0.86 0.87 1 0.17 6.9 0.083 0.18  
 [ reached getOption("max.print") -- omitted 24 rows ]  
  
 Item statistics   
 n raw.r std.r r.cor r.drop mean sd  
A1B3 15 0.6585 0.725 0.725 0.6326 0.933 0.26  
A2C3 15 0.7259 0.781 0.781 0.6959 0.867 0.35  
A3 15 0.0642 0.044 0.044 -0.0059 0.800 0.41  
A3B5C4 15 0.4650 0.446 0.446 0.4297 0.067 0.26  
A3B5E4 15 0.4420 0.436 0.436 0.3922 0.133 0.35  
A3C5D4 15 0.7055 0.666 0.666 0.6583 0.400 0.51  
A3C5E4 15 0.1061 0.109 0.109 0.0188 0.467 0.52  
A3D5E4 15 0.6293 0.609 0.609 0.5730 0.600 0.51  
A4B3D5 15 0.6102 0.564 0.564 0.5519 0.400 0.51  
B1E2 15 0.4511 0.497 0.497 0.4018 0.867 0.35  
 [ reached getOption("max.print") -- omitted 24 rows ]  
  
Non missing response frequency for each item  
 0 1 miss  
A1B3 0.07 0.93 0  
A2C3 0.13 0.87 0  
A3 0.20 0.80 0  
A3B5C4 0.93 0.07 0  
A3B5E4 0.87 0.13 0  
A3C5D4 0.60 0.40 0  
A3C5E4 0.53 0.47 0  
A3D5E4 0.40 0.60 0  
A4B3D5 0.60 0.40 0  
B1E2 0.13 0.87 0  
B2 0.07 0.93 0  
B2C5 0.07 0.93 0  
B3 0.07 0.93 0  
B4 0.27 0.73 0  
B4D5E3 0.60 0.40 0  
B4E3 0.20 0.80 0  
B5 0.60 0.40 0  
B5C4D3 0.53 0.47 0  
B5C4E3 0.73 0.27 0  
B5D4 0.60 0.40 0  
C1E4 0.20 0.80 0  
C2 0.07 0.93 0  
C3 0.13 0.87 0  
C3D2 0.33 0.67 0  
C4 0.27 0.73 0  
 [ reached getOption("max.print") -- omitted 9 rows ]

If you run this with warnings, it will note things like individual items negatively correlated with the total. The results show several results. The ‘raw alpha’ is based on covariances, and the ‘std.alpha’ is based on the correlations. These could differ if you incorporate measures that have very different scales, but should be fairly close if you have all of the same types of questions. These scores turn out to be very good for–actually close to the ``excellent’’ criterion of 0.9. Along with this, it reports G6 –an abbreviation for Guttman’s Lambda (). Guttman’s Lambda estimates the amount of variance in each item that can be accounted for by linear regression of all other items. The next column (average\_r) computes the average inter-item correlation. This is just an average of the off-diagonals of the correlation matrix. Here, this value is around 0.2. If this were really high, we would need a lot fewer items; because it is relatively low, this means that different questions are not completely predictive of anything else. The final column is a measure of signal-to-noise ratio, which you can read more on in the psych::alpha documentation.

Next, it looks at whether any specific items were especially influential, by examining these values of the item were dropped. These are all make a fairly small change, mostly.

## Item-total correlation

Finally, ``Item statistics’’ are shown. Here, the first column shows the correlation between each item and the total sum, the second computes a corrected version that corrects for item overlap and reliability, and r.drop correlates the item to the whole without that item. Here, we can start thinking about which items are really useful, and which are not, at least to the extent that they intend to measure a common factor. For example, B5C4D3- is actually negatively correlated with the total. This may indicate a good reason to remove the item from the test. However, if there are several factors, it might be a valuable index of another construct.

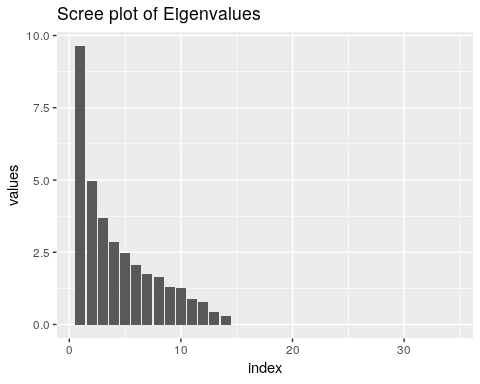
## Examining Eigenfactors and Principle Components (PCA)

Earlier, we stated that does not test whether you have a single construct, but rather assumes you do. To be confident in our analysis, we’d like to confirm that all the items fit together. The correlations are somewhat reassuring; but there were a lot of pairwise correlations that were negative and close to 0, which would happen if we had independent factors. To the extent that a specific set of questions are similar, you would expect them to all be correlated with one another. Principle Components analysis attempts to project the correlational structure onto a set of independent vectors using eigen decomposotion. This is related more generally to factor analysis, which will be covered later, and similar questions could be asked in a different way via clustering methods, but the typical approach used by researchers is eigen decomposition, also referred to as principal components analysis.

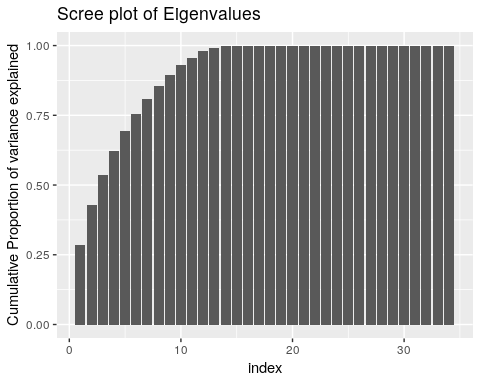
To the extent that a set of questions coheres, it should be well accounted for by a single factor. PCA identifies a proportion of variance accounted for, and if the proportion of variance for the first dimension is high and, as a rule of thumb, more than 3 times larger than the next, you might argue that the items show a strong coherence.

For a multi-attribute scale (i.e., personality big five) you’d expect multiple factors, but if you isolate just one dimension, you would hope that they can be explained by a single strong factor. We can do a simple factor analysis by doing eigen decomposition of the correlation matrix like this:

cor2 <- cor(items2)  
e <-eigen(cor2)  
vals <- data.frame(index=1:length(e$values),values=e$values)  
vals$prop <- cumsum(vals$values/sum(vals$values)) #Proportion explained  
ggplot(vals,aes(x=index,y=values))+geom\_bar(stat="identity") +   
 ggtitle("Scree plot of Eigenvalues")



ggplot(vals,aes(x=index,y=prop))+geom\_bar(stat="identity") +   
 ggtitle("Scree plot of Eigenvalues")+ylab("Cumulative Proportion of variance explained")



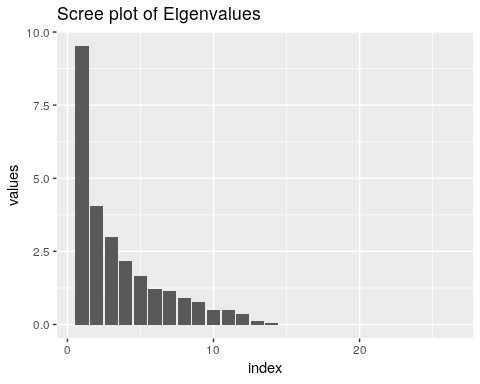
Looking at the ‘scree’ plot, this shows the proportion of variance accounted for by the most to least important dimension. The first dimension is about twice as great as the next, after which it trails off, so this is modestly-strong evidence for a single factor. A basic rule of thumb is that factors with eigenvalues greater than 1.0 are important, suggesting that there are 4-5 factors. The second figure shows cumulative proportion of variance explained (the sum of the eigenvalues is the total variance in the correlation matrix). Here, the results are again not compelling–only 25% of the variance is explained by one factor. These all point to there being multiple constructs that these measures are indexing.

Perhaps we can narrow down the questions to identify a single factor. Looking at the loadings on the first factor, we might look to eliminate values close to 0. It looks like questions 3, 7, 11, 18,23,26,28, and 30 are all relatively small (say less than .07 from 0). What happens if we select just the most strongly associated questions:

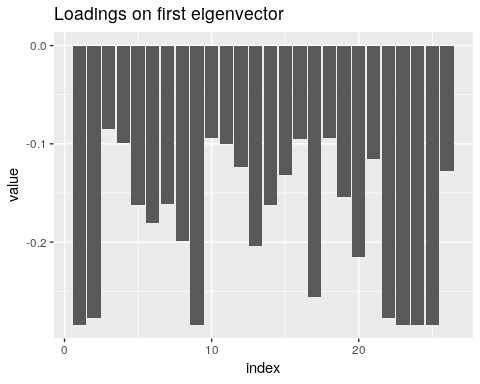
(1:34)[abs(e$vectors[,1])<.07]

[1] 3 7 11 18 23 26 28 30

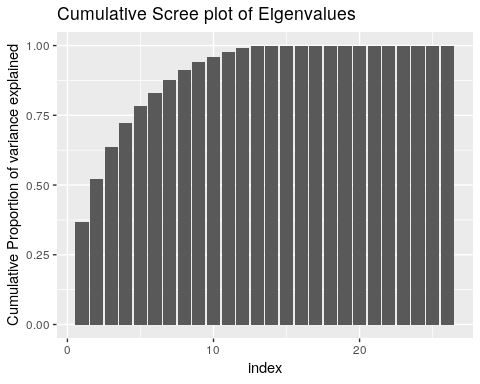
items3 <- items2[,abs(e$vectors[,1])>.07]  
e2 <- eigen(cor(items3))  
  
vals <- data.frame(index=1:length(e2$values),values=e2$values)  
vals$prop <- cumsum(vals$values/sum(vals$values)) #Proportion explained  
  
ggplot(vals,aes(x=index,y=values))+geom\_bar(stat="identity") +   
 ggtitle("Scree plot of Eigenvalues")



vec1 <- data.frame(index=1:nrow(e2$vectors),  
 value=e2$vectors[,1]  
 )  
ggplot(vec1,aes(x=index,y=value)) + geom\_bar(stat="identity")+ggtitle("Loadings on first eigenvector")



ggplot(vals,aes(x=index,y=prop))+geom\_bar(stat="identity") +   
 ggtitle("Cumulative Scree plot of Eigenvalues")+ylab("Cumulative Proportion of variance explained")



Let’s see what happens when we do the alpha:

alpha(items3)

Reliability analysis   
Call: alpha(x = items3)  
  
 raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
 0.9 0.92 1 0.3 11 0.037 0.66 0.22 0.27  
  
 lower alpha upper 95% confidence boundaries  
0.83 0.9 0.97   
  
 Reliability if an item is dropped:  
 raw\_alpha std.alpha G6(smc) average\_r S/N var.r med.r  
A1B3 0.89 0.91 1 0.29 10.0 0.065 0.24  
A2C3 0.89 0.91 1 0.28 9.9 0.068 0.22  
A3B5C4 0.90 0.92 1 0.31 11.1 0.072 0.29  
A3B5E4 0.90 0.92 1 0.31 11.0 0.072 0.30  
A3C5D4 0.90 0.91 1 0.30 10.6 0.072 0.27  
A3D5E4 0.90 0.91 1 0.30 10.5 0.073 0.24  
A4B3D5 0.90 0.91 1 0.30 10.6 0.073 0.27  
B1E2 0.90 0.91 1 0.30 10.6 0.070 0.27  
B2C5 0.89 0.91 1 0.29 10.0 0.065 0.24  
B3 0.90 0.92 1 0.31 11.1 0.066 0.29  
 [ reached getOption("max.print") -- omitted 16 rows ]  
  
 Item statistics   
 n raw.r std.r r.cor r.drop mean sd  
A1B3 15 0.75 0.79 0.79 0.73 0.933 0.26  
A2C3 15 0.81 0.85 0.85 0.79 0.867 0.35  
A3B5C4 15 0.38 0.36 0.36 0.34 0.067 0.26  
A3B5E4 15 0.41 0.40 0.40 0.36 0.133 0.35  
A3C5D4 15 0.61 0.56 0.56 0.55 0.400 0.51  
A3D5E4 15 0.61 0.59 0.59 0.55 0.600 0.51  
A4B3D5 15 0.61 0.57 0.57 0.55 0.400 0.51  
B1E2 15 0.52 0.56 0.56 0.48 0.867 0.35  
B2C5 15 0.75 0.79 0.79 0.73 0.933 0.26  
B3 15 0.36 0.36 0.36 0.32 0.933 0.26  
 [ reached getOption("max.print") -- omitted 16 rows ]  
  
Non missing response frequency for each item  
 0 1 miss  
A1B3 0.07 0.93 0  
A2C3 0.13 0.87 0  
A3B5C4 0.93 0.07 0  
A3B5E4 0.87 0.13 0  
A3C5D4 0.60 0.40 0  
A3D5E4 0.40 0.60 0  
A4B3D5 0.60 0.40 0  
B1E2 0.13 0.87 0  
B2C5 0.07 0.93 0  
B3 0.07 0.93 0  
B4 0.27 0.73 0  
B4D5E3 0.60 0.40 0  
B4E3 0.20 0.80 0  
B5 0.60 0.40 0  
B5C4E3 0.73 0.27 0  
B5D4 0.60 0.40 0  
C1E4 0.20 0.80 0  
C2 0.07 0.93 0  
C3D2 0.33 0.67 0  
C4 0.27 0.73 0  
D1E5 0.40 0.60 0  
D4 0.13 0.87 0  
E2 0.07 0.93 0  
E3 0.07 0.93 0  
E4 0.07 0.93 0  
 [ reached getOption("max.print") -- omitted 1 row ]

Now, alpha went up, as did all the measures of consistency. This is still weak evidence for a single factor, but also remember we are basing this on a small sample size, and before making any decisions we’d want to collect a lot more data.

## Alternate measures of consistency

As discussed before (see Sijtsma, 2009), modern statisticians suggest that is fairly useless, and should only be used hand-in-hand with other better measures, including one called glb (greatest lower bound). the psych package provides several alternative measures. Note that for some of these, we don’t have enough observations to do a complete analysis. First, running glb(items3) will compute a set of measures (guttman, tenbarg, glb, and glb.fa).

g <-glb(items3)  
  
g$beta #worst split-half reliability

[1] 0.4669289

g$alpha.pc #estimate of alpha

[1] 0.8951595

g$glb.max #best alpha

[1] 0.9422672

g$glb.IC #greatest lower bound using ICLUST clustering

[1] 0.9047976

g$glb.Km #greatest lower bound using Kmeans clustering

[1] 0.8137413

g$glb.Fa #greatest lower bound using factor analysis

[1] 0.9422672

g$r.smc #squared multiple correlation

[1] 1

g$tenberge #the first value is alpha

$mu0  
[1] 0.9167904  
  
$mu1  
[1] 0.9286011  
  
$mu2  
[1] 0.929218  
  
$mu3  
[1] 0.9292724

Now, the beta score shows us that possibly our coherence is .47. This converges with our examination of the factors–there might be more than one reasonable factor in the data set.

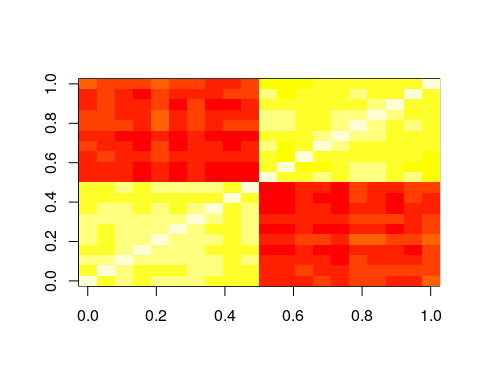
As a brief test, what happen when our data have two very strong factors in them? Suppose we have 10 items from each of two factors

set.seed(102)  
base1 <- rnorm(10)\*3  
base2 <- rnorm(10)\*3  
people1 <- rnorm(50)  
people2 <- rnorm(50)  
  
cor(base1,base2)

[1] -0.1387837

These are slightly negatively correlated.

noise <- .5  
  
set1 <- outer(people1,base1,"+" ) +matrix(rnorm(500,sd=noise),nrow=50)  
set2 <- outer(people2,base2,"+" ) +matrix(rnorm(500,sd=noise),nrow=50)  
  
data <-as.data.frame(cbind(set1,set2))  
image(cor(data))



e <- eigen(cor(data))  
e$values

[1] 10.02706648 6.93182253 0.41403628 0.34552463 0.30053502  
 [6] 0.27505086 0.25045743 0.22005759 0.19291691 0.17907262  
[11] 0.14719890 0.13298576 0.11112113 0.09858208 0.09272425  
[16] 0.08485916 0.05798296 0.05511698 0.04466288 0.03822552

alpha(data)

Reliability analysis   
Call: alpha(x = data)  
  
 raw\_alpha std.alpha G6(smc) average\_r S/N ase mean sd median\_r  
 0.95 0.95 0.99 0.47 18 0.013 0.92 0.81 0.26  
  
 lower alpha upper 95% confidence boundaries  
0.92 0.95 0.97   
  
 Reliability if an item is dropped:  
 raw\_alpha std.alpha G6(smc) average\_r S/N alpha se var.r med.r  
V1 0.94 0.94 0.99 0.47 17 0.014 0.12 0.25  
V2 0.94 0.94 0.99 0.47 17 0.014 0.12 0.25  
V3 0.94 0.94 0.99 0.47 17 0.014 0.12 0.26  
V4 0.94 0.95 0.99 0.48 17 0.014 0.11 0.26  
V5 0.94 0.94 0.99 0.47 17 0.014 0.12 0.25  
V6 0.94 0.95 0.99 0.47 17 0.014 0.11 0.26  
V7 0.94 0.94 0.99 0.47 17 0.014 0.12 0.26  
V8 0.94 0.95 0.99 0.48 17 0.013 0.11 0.26  
V9 0.94 0.95 0.99 0.48 17 0.013 0.12 0.26  
 [ reached getOption("max.print") -- omitted 11 rows ]  
  
 Item statistics   
 n raw.r std.r r.cor r.drop mean sd  
V1 50 0.75 0.74 0.73 0.71 0.83 1.20  
V2 50 0.76 0.75 0.74 0.72 2.65 1.18  
V3 50 0.74 0.72 0.72 0.70 -3.91 1.21  
V4 50 0.70 0.69 0.68 0.66 6.21 1.17  
V5 50 0.78 0.77 0.77 0.75 3.92 1.17  
V6 50 0.70 0.69 0.69 0.67 3.95 1.13  
V7 50 0.76 0.75 0.74 0.73 2.92 1.17  
V8 50 0.70 0.68 0.68 0.65 1.01 1.20  
V9 50 0.69 0.68 0.67 0.65 1.84 1.11  
V10 50 0.73 0.71 0.71 0.69 6.07 1.23  
 [ reached getOption("max.print") -- omitted 10 rows ]

glb(data)

$beta  
[1] 0.3090444  
  
$beta.factor  
[1] 0.3090444  
  
$alpha.pc  
[1] 0.9002699  
  
$glb.max  
[1] 0.9658362  
  
$glb.IC  
[1] 0.9658362  
  
$glb.Km  
[1] 0.3090444  
  
$glb.Fa  
[1] 0.9582077  
  
$r.smc  
[1] 0.9875206  
  
$tenberge  
$tenberge$mu0  
[1] 0.947368  
  
$tenberge$mu1  
[1] 0.9583356  
  
$tenberge$mu2  
[1] 0.9589135  
  
$tenberge$mu3  
[1] 0.9589372  
  
  
$keys  
 IC1 IC2 ICr1 ICr2 K1 K2 F1 F2 f1 f2  
V1 1 0 1 0 0 1 1 0 1 0  
V2 1 0 1 0 0 1 0 1 1 0  
V3 1 0 1 0 0 1 0 1 1 0  
V4 1 0 1 0 0 1 0 1 1 0  
V5 1 0 1 0 0 1 0 1 1 0  
V6 1 0 0 1 0 1 1 0 1 0  
V7 1 0 1 0 0 1 1 0 1 0  
 [ reached getOption("max.print") -- omitted 13 rows ]

glb(set1)

$beta  
[1] 0.9707496  
  
$beta.factor  
[1] NaN  
  
$alpha.pc  
[1] 0.883662  
  
$glb.max  
[1] 0.9873093  
  
$glb.IC  
[1] 0.9805451  
  
$glb.Km  
[1] 0.9873093  
  
$glb.Fa  
[1] 0.9826313  
  
$r.smc  
[1] 0.9823743  
  
$tenberge  
$tenberge$mu0  
[1] 0.981806  
  
$tenberge$mu1  
[1] 0.9818698  
  
$tenberge$mu2  
[1] 0.9818825  
  
$tenberge$mu3  
[1] 0.981885  
  
  
$keys  
 IC1 IC2 ICr1 ICr2 K1 K2 F1 F2 f1 f2  
V1 0 1 1 0 0 1 0 1 1 0  
V2 0 1 1 0 0 1 0 1 1 0  
V3 0 1 1 0 0 1 1 0 1 0  
V4 0 1 1 0 0 1 1 0 1 0  
V5 1 0 1 0 0 1 0 1 1 0  
V6 0 1 0 1 1 0 0 1 1 0  
V7 0 1 1 0 1 0 1 0 1 0  
 [ reached getOption("max.print") -- omitted 3 rows ]

Now, we can see that when we don’t have a single factor, we can still get high coefficient alpha, but some of the other measures produced by glb will fail.

## Specifying reverse coded elements

Many times, a scale will involve multiple questions, with some reverse coded. That is, a high score on one question (e.g., How much do you like pancakes?) is correlated with a low score on another question (e.g, How much do you hate pancakes?). These are referred to as ‘reverse key’, and the The psych::alpha and glb functions have different ways of handling these.

* psych::alpha will check whether the items are all positively related and give a message if not

psych::alpha(data)  
Warning message:  
In psych::alpha(dimC) :  
 Some items were negatively correlated with the total scale and probably   
should be reversed.   
To do this, run the function again with the 'check.keys=TRUE' option

* psych::alpha will let you specify the names of the variable to reverse code using the keys argumen

psych::alpha(data,key=c("q1","q3"))

* psych::alpha will also accept a vector of column indices:

psych::alpha(data,key=c(1,3))

* pysch::alpha will use its best guess of coding if you use check.keys=T

psych::alpha(data,check.keys=T)

* psych::glb will not check automatically and assume that the items are all positive, but provides a $keys output that will help identify different groups

$keys  
 IC1 IC2 ICr1 ICr2 K1 K2 F1 F2 f1 f2  
Q3 1 0 0 1 0 1 0 1 1 0  
Q8 0 1 0 1 0 1 1 0 0 1  
Q13 1 0 1 0 1 0 1 0 1 0  
Q18 0 1 1 0 0 1 1 0 0 1  
Q23 0 1 1 0 1 0 0 1 0 1  
Q28 1 0 1 0 1 0 0 1 1 0  
Q33 1 0 1 0 0 1 1 0 1 0  
Q38 1 0 1 0 0 1 1 0 1 0  
Q43 0 1 1 0 1 0 0 1 0 1

* psych:splitHalf will accept the check.keys=T argument, which will automatically recode

psych::splitHalf(dimC,check.keys=T)  
Split half reliabilities   
Call: psych::splitHalf(r = dimC, check.keys = T)  
  
Maximum split half reliability (lambda 4) = 0.82  
Guttman lambda 6 = 0.77  
Average split half reliability = 0.75  
Guttman lambda 3 (alpha) = 0.77  
Minimum split half reliability (beta) = 0.62  
Average interitem r = 0.27 with median = 0.26Warning message:  
In psych::splitHalf(dimC, check.keys = T) :  
 Some items were negatively correlated with total scale and were automatically reversed.

* psych::glb will not check automatically and assume that the items are all positive, but provides a $keys output that will help identify different groups

$keys  
 IC1 IC2 ICr1 ICr2 K1 K2 F1 F2 f1 f2  
Q3 1 0 0 1 0 1 0 1 1 0  
Q8 0 1 0 1 0 1 1 0 0 1  
Q13 1 0 1 0 1 0 1 0 1 0  
Q18 0 1 1 0 0 1 1 0 0 1  
Q23 0 1 1 0 1 0 0 1 0 1  
Q28 1 0 1 0 1 0 0 1 1 0  
Q33 1 0 1 0 0 1 1 0 1 0  
Q38 1 0 1 0 0 1 1 0 1 0  
Q43 0 1 1 0 1 0 0 1 0 1

* psych:splitHalf will accept the check.keys=T argument, which will automatically recode and give a warning:

psych::splitHalf(dimC,check.keys=T)  
Split half reliabilities   
Call: psych::splitHalf(r = dimC, check.keys = T)  
  
Maximum split half reliability (lambda 4) = 0.82  
Guttman lambda 6 = 0.77  
Average split half reliability = 0.75  
Guttman lambda 3 (alpha) = 0.77  
Minimum split half reliability (beta) = 0.62  
Average interitem r = 0.27 with median = 0.26Warning message:  
In psych::splitHalf(dimC, check.keys = T) :  
 Some items were negatively correlated with total scale and were automatically reversed.

* psych::glb will accept a coding vector that is -1/1/0, which differs from psych::alpha:

glb(data,key=c(-1,1,-1,1,1,1))

You can use a -1/1 vector in psych::alpha with a coding like this:

psych::alpha(data,key=(1:6)[-1 == c(-1,1,-1,1,1,1)])

## Exercise.

Load the ability data set from the psych package, and compute coefficient alpha and the other alternative measures.

help(ability)

# Split-half Reliability and correlation

There are two sensible ways in which you might compute a split-half correlation: splitting your items into two bins and correlating across the participants, and splitting participants into two bins and correlating across items. When people discuss split-half reliability, they typically referring to the process of splitting your test into two sub-tests (usually randomly) and finding the correlation. Splitting your participants into two groups and computing the correlation could be used as a way to bootstrap confidence intervals of your correlation coefficient, especially when you are unsure of whether you satisfy assumptions of the tests of statistical inference. Furthermore, if you collect enough data (millions of subjects), you’d expect essentially the same values when splitting your participants into groups. Typically, split-half approaches refer to splitting your items or questions into two groups.

# Split-half correlation: splitting questions or items

Oftentimes, there is a natural way to split your questions into two equivalent sub-tests, either for some specific reason (first versus second half) or at random. The rationale for this is often to establish a sort of test-retest reliability within one sample. Some of the psychometric methods will do this for us automatically.

## Split-half Measures

Finally, we can compute reliabilities for split-half comparisons. The splitHalf function will find all split-half comparisons–dividing the test into two sub-tests, computing means, and correlating results. For small numbers of items, it does all possible splits. For more items (> 16), it will sample from the possible splits. Remember that this is sort of what Cronbach’s is supposed to be computing anyway. Let’s look at split-half for the first 7 questions:

s <- splitHalf(items3[,1:7])  
s

Split half reliabilities   
Call: splitHalf(r = items3[, 1:7])  
  
Maximum split half reliability (lambda 4) = 0.88  
Guttman lambda 6 = 0.84  
Average split half reliability = 0.76  
Guttman lambda 3 (alpha) = 0.79  
Minimum split half reliability (beta) = 0.57  
Average interitem r = 0.35 with median = 0.33

Here, not that the measures for this subset are typically very good (.76+). The beta value–which is the worst-case scenario–is the worst correspondence that was found, and is still somewhat reasonable. What if we had taken 10 items instead?

splitHalf(items3[,1:10])

Split half reliabilities   
Call: splitHalf(r = items3[, 1:10])  
  
Maximum split half reliability (lambda 4) = 0.96  
Guttman lambda 6 = 0.93  
Average split half reliability = 0.81  
Guttman lambda 3 (alpha) = 0.82  
Minimum split half reliability (beta) = 0.54  
Average interitem r = 0.31 with median = 0.32

Notice the the average gets better, but the worst case actually goes down a bit–having more pairings gives more opportunities for a bad correspondence.

You can learn more about the specific statistics reported here using the R documentation–the psych package has very good references and explanations.

Look at the two-factor data set:

glb(data)

$beta  
[1] 0.3090444  
  
$beta.factor  
[1] 0.3090444  
  
$alpha.pc  
[1] 0.9002699  
  
$glb.max  
[1] 0.9658362  
  
$glb.IC  
[1] 0.9658362  
  
$glb.Km  
[1] 0.3090444  
  
$glb.Fa  
[1] 0.9582077  
  
$r.smc  
[1] 0.9875206  
  
$tenberge  
$tenberge$mu0  
[1] 0.947368  
  
$tenberge$mu1  
[1] 0.9583356  
  
$tenberge$mu2  
[1] 0.9589135  
  
$tenberge$mu3  
[1] 0.9589372  
  
  
$keys  
 IC1 IC2 ICr1 ICr2 K1 K2 F1 F2 f1 f2  
V1 1 0 1 0 1 0 1 0 1 0  
V2 1 0 1 0 1 0 0 1 1 0  
V3 1 0 1 0 1 0 0 1 1 0  
V4 1 0 1 0 1 0 0 1 1 0  
V5 1 0 1 0 1 0 0 1 1 0  
V6 1 0 0 1 1 0 1 0 1 0  
V7 1 0 1 0 1 0 1 0 1 0  
 [ reached getOption("max.print") -- omitted 13 rows ]

## Brown-Spearman Prophecy Formula

Typically, the split-half correlation is thought to be biased to under-estimate the true relationship between the two halves. People will sometimes apply the Brown-Spearman prediction formula, which estimates the expected reliability of a test if its length is changed. So, if you have a small test having reliability , then if you were to increase the length of the test by factor , the new test would have reliability .

For a split-half correlation, you would estimate this as

`

If you complete a split-half correlation of .5, you could adjust this to be . However, especially for small tests and small subject populations, this seems rash, because you probably don’t have a very good estimate of the correlation. We can look at this via simulation:

set.seed(100)  
logit <- function(x){1/(1+1/exp(x))}  
  
simulate <- function(numsubs,numitems)  
{  
   
 submeans <- rnorm(numsubs,sd=10)  
 itemmeans <- rnorm(numitems,sd=5)  
  
 simdat1 <- (logit(outer(submeans,itemmeans)) >matrix(runif(numsubs\*numitems),  
 nrow=numsubs,ncol=numitems))+0  
  
simdat2 <- (logit(outer(submeans,itemmeans)) >matrix(runif(numsubs\*numitems),  
 nrow=numsubs,ncol=numitems))+0  
realcor <- cor(rowSums(simdat1),rowSums(simdat2))  
halfcor <- cor(rowSums(simdat1[,2\*1:floor(numitems/2)]),  
 rowSums(simdat1[,2\*1:floor(numitems/2)-1]))  
##adjusted  
adjusted <- (2\*halfcor)/(1+halfcor)  
 c(realcor,halfcor,adjusted)  
}  
newvals <- matrix(0,nrow=100,ncol=3)  
for(i in 1:100)  
 {  
 newvals[i,] <- simulate(1000,20)  
 }  
colMeans(newvals,na.rm=T)

[1] 0.6518145 0.0422957 -2.7434579

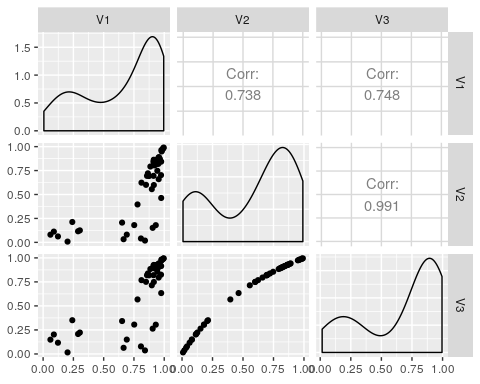
newvals[newvals[,2]<0,2] <- NA  
newvals[newvals[,3]<0,3] <- NA  
  
cor(newvals,use="pairwise.complete")

[,1] [,2] [,3]  
[1,] 1.0000000 0.7379033 0.7481689  
[2,] 0.7379033 1.0000000 0.9907501  
[3,] 0.7481689 0.9907501 1.0000000

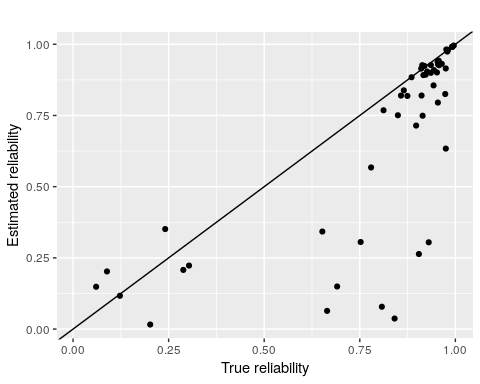
head(newvals)

[,1] [,2] [,3]  
[1,] 0.19215909 NA NA  
[2,] 0.19242084 NA NA  
[3,] 0.82532920 NA NA  
[4,] 0.18154840 NA NA  
[5,] 0.20179081 0.008037462 0.01594675  
[6,] 0.08867742 0.112542949 0.20231659

tmp <- as.data.frame(newvals)  
ggpairs(tmp)



ggplot(tmp,aes(x=V1,y=V3))+geom\_point() + geom\_abline(intercept=0,slope=1) +  
 ggtitle("")+ylab("Estimated reliability")+xlab("True reliability")



Here, the first number indicates the reliability of the test, built into the simulation. The second number is the estimate for the split-half, and the third number is the adjusted value. If we ignore the cases where the split-half is negative (which happens fairly often), there is a reasonable correlation between the actual estimate of test-retest and the split-half. Even for 1000 participants, in this case.

# Categorical Agreement: Cohen’s (kappa)

When you have several nominally-coded sets of scores, correlations won’t work anymore. Instead, we use need another measure of agreement. The standard measure is Cohen’s . This is most useful for comparing two specific data sets of categorical responses. The library can compute using the function.

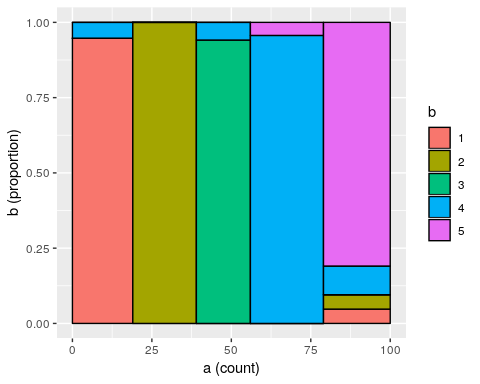
##courtesy http://stackoverflow.com/questions/19233365/how-to-create-a-marimekko-mosaic-plot-in-ggplot2  
## see library(ggmosiac) for an alternative that integrates better with ggplot2  
  
makeplot\_mosaic <- function(data, x, y, ...){  
 xvar <- deparse(substitute(x))  
 yvar <- deparse(substitute(y))  
 mydata <- data[c(xvar, yvar)];  
 mytable <- table(mydata);  
 widths <- c(0, cumsum(apply(mytable, 1, sum)));  
 heights <- apply(mytable, 1, function(x){c(0, cumsum(x/sum(x)))});  
  
 alldata <- data.frame();  
 allnames <- data.frame();  
 for(i in 1:nrow(mytable)){  
 for(j in 1:ncol(mytable)){  
 alldata <- rbind(alldata, c(widths[i], widths[i+1], heights[j, i], heights[j+1, i]));  
 }  
 }  
 colnames(alldata) <- c("xmin", "xmax", "ymin", "ymax")  
  
 alldata[[xvar]] <- rep(dimnames(mytable)[[1]],rep(ncol(mytable), nrow(mytable)));  
 alldata[[yvar]] <- rep(dimnames(mytable)[[2]],nrow(mytable));  
  
 ggplot(alldata, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax)) +   
 geom\_rect(color="black", aes\_string(fill=yvar)) +  
 xlab(paste(xvar, "(count)")) + ylab(paste(yvar, "(proportion)"));  
}  
library(psych)  
##What if we have high agreement:   
a <- sample(1:5,100,replace=T)  
b <- a  
  
cohen.kappa(cbind(a,b))

Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)  
  
Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries   
 lower estimate upper  
unweighted kappa 1 1 1  
weighted kappa 1 1 1  
  
 Number of subjects = 100

#replace 10 elements from b  
b[sample(1:100,10)] <- sample(1:5,10,replace=T)  
table(a,b)

b  
a 1 2 3 4 5  
 1 18 0 0 1 0  
 2 0 20 0 0 0  
 3 0 0 16 1 0  
 4 0 0 0 22 1  
 5 1 1 0 2 17

tmp <- data.frame(a=a,b=b)  
makeplot\_mosaic(tmp,a,b)



cohen.kappa(cbind(a,b))

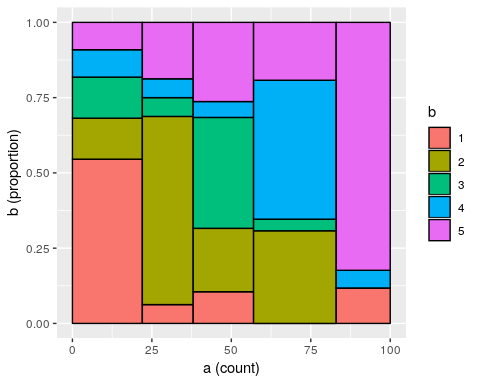
Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)  
  
Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries   
 lower estimate upper  
unweighted kappa 0.85 0.91 0.97  
weighted kappa 0.81 0.90 1.00  
  
 Number of subjects = 100

What if we add more noise:

a <- sample(1:5,100,replace=T)  
b <- a  
#replace 50 elements from b  
b[sample(1:100,50)] <- sample(1:5,50,replace=T)  
table(a,b)

b  
a 1 2 3 4 5  
 1 12 3 3 2 2  
 2 1 10 1 1 3  
 3 2 4 7 1 5  
 4 0 8 1 12 5  
 5 2 0 0 1 14

#mosaicplot(table(a,b),col=1:5)  
  
tmp <- data.frame(a=a,b=b)  
makeplot\_mosaic(tmp,a,b)



cohen.kappa(cbind(a,b))

Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)  
  
Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries   
 lower estimate upper  
unweighted kappa 0.32 0.44 0.56  
weighted kappa 0.34 0.52 0.70  
  
 Number of subjects = 100

Sometimes, you might determine that some categories are more similar than others. Thus, you may want to score partial agreement. A weighted version of permits this. So, suppose that our five categories represent ways of coding a video for signs of boredom

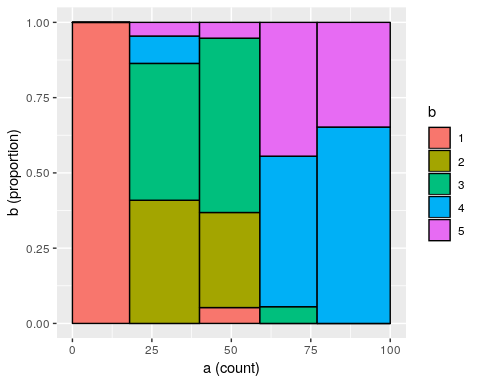
1. rolling eyes
2. yawning
3. falling asleep
4. checking email on phone
5. texting on phone

Here, 2 and 3 are sort of similar, as are 4 and 5. Two people coding video might mistake these for one another, and it might be reasonable. So, we can use a weighted

a <- sample(1:5,100,replace=T)  
b <- a  
b[sample(1:100,10)] <- sample(1:5,10,replace=T)  
  
#replace 50 elements from b  
b45 <- b==4|b==5  
b[b45] <- sample(4:5,sum(b45),replace=T)  
b23 <- b==2|b==3  
b[b23] <- sample(2:3,sum(b23),replace=T)  
  
weights <- matrix(c(0,1,1,1,1,  
 1,0,.5,1,1,  
 1,.5,0,1,1,  
 1,1,1,0,.5,  
 1,1,1,.5,0),nrow=5,ncol=5)  
table(a,b)

b  
a 1 2 3 4 5  
 1 18 0 0 0 0  
 2 0 9 10 2 1  
 3 1 6 11 0 1  
 4 0 0 1 9 8  
 5 0 0 0 15 8

tmp <- data.frame(a=a,b=b)  
makeplot\_mosaic(tmp,a,b)



cohen.kappa(cbind(a,b))

Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)  
  
Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries   
 lower estimate upper  
unweighted kappa 0.32 0.44 0.56  
weighted kappa 0.75 0.83 0.91  
  
 Number of subjects = 100

cohen.kappa(cbind(a,b),weights)

Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels)  
  
Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries   
 lower estimate upper  
unweighted kappa 0.32 0.44 0.56  
weighted kappa 0.64 0.64 0.64  
  
 Number of subjects = 100

Notice how the first value (unweighted) does not change when we add weights, but the second weighted does change. By default, the weighted version assumes the off-diagonals are ‘’quadratically’ weighted, meaning that the farther apart the ratnigs are, the worse the penalty. This would be appropriate for likert-style ratings, but not for categorical ratings. For numerical/ordinal scales, the documentation claims is similar to ICC.

## Exercise.

Have two people rate the following words into emotional categories: happy, sad, excited, bored, disgusted (HSEBD)

* horse
* potato
* dog
* umbrella
* coffee
* tennis
* butter
* unicorn
* eggs
* books
* rectangle
* twitter
* water
* soup
* crows
* Canada
* tomato
* nachos
* gravy
* chair

Compute kappa on your pairwise ratings.