**COMPARING TWO KENDALL’S TAU CORRELATION COEFFICIENTS**

Jason R. Finley

Feb 4th, 02010

see R code I made: CCseJRFcompare4.r

main references:

Woods 2007, 2009

Cliff & Charlin 1991, Cliff 1996a (book), Cliff 1996b

Long & Cliff, 1997

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(I) INTRO**

The typical correlation measurement used is Pearson’s r. Kendall’s tau (of which there are a few variants [a, b, c] and I focus on a and b here) is a rank-based (i.e., ordinal) measure of association between two variables. It is a useful alternative to r when:

-ordinal data

-non-normal distributions

-non-linear monotonic bivariate relationship (e.g., perfect exponential relationship)

-also, it’s resistant to outliers.

Spearman’s rho and Goodman & Kruskal’s gamma (the latter often used in metamemory research) are two other major alternatives. Spearman’s rho is just Pearson’s r performed on ranked data (i.e., you convert each data point within a variable into a rank order, usually with 1 assigned to the lowest value, 2 to the second-lowest etc.). This makes it better than r when the assumptions for r are violated. Rho was preferred to tau before everyone had supercomputers; now, tau should probably be generally preferred (I’m not clear on why). Gamma seems to be as useful as tau as long as there are no ties in the ranked data. When there are ties, gamma disregards those data, as does tau-a. Thus, if there are many ties, tau-b should be used, as it is more conservative. Tau-c is another version of tau that accounts for ties, but it’s not preferred (and I forget why).

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(II) CALCULATING TAU-A and TAU-B**

N = number of subjects

C = number of concordances (may also be referred to as P)

D = number of discordances (may also be referred to as Q)

You have 2 variables, call them X and Y. You should have NO MISSING CELLS. So exclude from analysis any subjects who are missing either an X or a Y value. You’ll also need an index/ID number for each row. So, something like:

SubNum X Y

1 45 16

2 12 34

3 11 2

4 12 15

i = index1

j = index2

Go through all possible pairings of subjects (for i = 1 to N { for j = 1 to N { … exclude pairings where i = j) and for each ij :

if ((Xi > Xj & Yi > Yj) OR (Xi < Xj & Yi < Yj)) that’s a CONCORDANCE. (that is, if X and Y “agree” on which subject should be ranked higher, which also means that the sign of the (Xi-Xj) difference is the same as the sign of the (Yi-Yj) difference)

if X and Y “disagree” on which subject should be ranked higher, then that’s a DISCORDANCE.

if Xi = Xj or Yi = Yj (or both), then that’s a TIE and it doesn’t count as either a concordance or discordance.

Tau-a (aka txy) is calculated as follows (from Woods, 2007, table 1):



where:

Tx = # of ties on just X

Ty = # of ties on just Y

Txy = # of ties on both X and Y

Tau-a does not take into consideration ties. If there are many ties, as will be the case if your data are ordinal (e.g., you have an integer 1-7 for X and Y for each subject), then this is a problem. That’s what Tau-b is for. Tau-b handles ties. it is calculated as follows (from Woods, 2007, table 1):



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(III) CALCULATING STANDARD ERRORS FOR TAU-A**

(note: be sure to be careful and not lose track of variances vs SDs/SEs)

(another note: I’m use the lowercase t to represent the sample Tau)

So far so good. Now the tricky thing is to get standard errors (SE) for individual Tau-a and Tau-b, and for comparing two Tau-as or two Tau-bs. Once we’ve got those, we can do null hypothesis significance tests (e.g., test a Tau against null hypothesis value of zero, or test a difference of two Taus against zero) and/or make confidence intervals.

Once you get a variance here, you’ll just take the square root and that will give you both the standard deviation (SD) and the standard error (SE). Here SD=SE. There’s no need to correct the SD like we would with the mean (i.e., SE = SD/SQRT(N-1)). I’m not sure exactly why, but I think it’s because Tau, and the SD we estimate from it, is a random variable and unbiased estimator of the population values… or maybe that the estimated variances have already been corrected to be unbiased… Anyway, Woods 2007 specifically notes that: “The formulas cited in this paragraph have been simpliﬁed so that SE = SQRT(variance); division by N is unnecessary.”

There are two major approaches to deriving the variance of a tau: randomization, and parametric (and two different methods of the latter: unbiased, and consistent). For now I’m just going to cover doing this all for tau-a. I’ll get back to tau-b.

(A) Randomization approach

This approach assumes that X and Y are independent, and thus can really only be used to test that specific null hypothesis of independence. It also takes advantage of the fact that tau is asymptotically normal (approaches normality with large sample sizes).

Here’s the formula

var(txy) = (4N + 10) / (9N(N-1))

(B) Parametric approach

This approach does NOT assumes independence between X and Y. It estimates the population variance of tau from the sample. There are two different methods within this approach (most clearly differentiated by Woods, 2007; see also Cliff 1996a): *unbiased* and *consistent*. The consistent estimates are recommended over the unbiased estimates by Long & Cliff (1997), Cliff (1996a), and Woods (2007).

tijxy: for a given pair of subjects (exclude i = j), this value is 1 if concordance, -1 if discordances, and 0 if tied on X or Y or both.

(B.1) Unbiased

The unbiased variance for tau-a is calculated as follows (from Woods, 2007, equation 7) [note that there are two major terms in it that need to be further defined]:



Note: if the estimated value from that formula ends up being ≤ 0, then you should instead use the following (Cliff, 1996a, p 61):



Now to define the two as-yet-undefined components of the formula for the unbiased variance of tau-a. The below formulas are also from Woods 2007:

(a)  


(b)  


Ha! Wait, what’s ti.xy? Okay, that is:  
 

So you can see, you’re going to have to loop through all possible pairings of subjects, excluding i=j, just like you originally did to calculate tau in the first place, so you can calculate tijxy for each valid pairing, and then you’ll have to compute ti.xy for each subject (i.e., each possible value of i) by summing tijxy over all other subjects and dviding the sum by N-1, then also compute the squared deviation of ti.xy from txy (aka the tau-a value you calculated) and sum those squared deviations and divide by N which gives you (b) above, then to get (a) you’ll have to compute a squared deviation of each tijxy from txy (tau-a) and sum all those and divide by N(N-1). THEN you can put together the pieces to calculate the unbiased variance of tau-a. This whole procedure will either require a bunch of columns and array formulas in Excel, or for loops in R (or MatLab or whatever).

Remember, once you’ve got the variance for a tau, take the square root and that’s your SE.

(B.2) Consistent

These are the ones you actually want to use.



**\*\*\*NOTE\*\*\*: major differences** between this formula and the unbiased formula given earlier:

(N-2) in numerator instead of (N-1)

PLUS SIGN in numerator instead of minus sign

denominator is N(N-1) instead of (N-2)(N-3)

Also, the components in the numerator, (a) and (b), are different in that another 1 is subtracted from the denominator. I use notation a little different from Woods 2007.

(a)  
 

(b)  
   
 and ti.xy is again:



Great! Now you’ve got the consistent variance estimate of a tau-a. You can use that to do a z-test of tau-a against the null hypothesis of zero. Why z instead of t? I’m not sure, but that’s how Cliff and Woods do it, and that’s how it’s done for r correlations too, I think. z = (txy-0)/SEtxy

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(IV) CALCULATING STANDARD ERRORS FOR TAU-B**

This gets difficult. It looks like the variance we get for tau-b is asymptotic (i.e., randomization?) rather than unbiased or consistent. No one has yet apparently taken on the task of figuring out an unbiased or consistent estimate of variance for tau-b. One thing is for sure though: in calculating the SE for tau-b, we want to use the CONSISTENT estimates of tau-a variances wherever variances are needed. We’re also going to need covariances, and I’m only going to bother defining the consistent versions of those for now (not the unbiased versions).

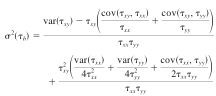
So, we already have txy (aka tau-a), and can get the consistent variance for that. Now we’re also going to need txx and tyy and the variances for those. From Woods (2007): txx is “the probability that a pair is not tied on X”. Here’s the formula for txx:



tijx: is 0 if the pair i,j is tied on X, and 1 otherwise.

Remember to exclude “pairs” where i=j (since that would just be comparing a single subject to him/herself, so of course it’s going to be a tie!). Tyy is computed the same way, just replace all the Xs in the above formula with Ys, and tiyy is 0 if a pair ties on Y and 1 otherwise.

Great, now we’re ready for the formula for the variance of tau-b, from Woods (2007, equation 14):



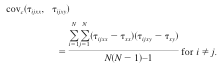
Criminey. Note that this formula is broken up over two lines. It also might make calculation a little easier to separately calculate the two major components. Quick note: when you see  that just means you square the txy value you’ve got. Now, the main things we’re missing in order to compute the above are the covariances of two tau-a values. Woods 2007 also gives unbiased formulas for these (Appendix A). I’m just going to give the consistent ones here.

Consistent covariance between two tau-a values:



Again, there are two components that need to be defined.

(a)  
 

(b)  
 

So basically, all the same kinda crap you had to do for getting the consistent variance of tau-a, you’ll have to do to get these covariances too. Note that the above 3 formulas give you the covariance between txx and txy. The order doesn’t matter here. Also, the same formulas can be used to get covariance between any other two tau-a values, for example: txx and tyy, or tyy and txy.

Do all this and you’ll get your estimated SE for a tau-b, based on the consistent variances of tau-a.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(V) CALCULATING STANDARD ERRORS FOR DIFFERENCES BETWEEN TAUS**

Okay, major leagues here.

I’m not sure whether to use z or t for the significance test. Cliff (1996b, p 341), says to use z-test to compare taus. Other main option would be t-test, but I'm not sure what appropriate df would be. Maybe: for independent: df=N1+N2-4, for dependent with two taus drawing on same variable: df=N-3, and for dependent with two taus drawing on different variables: df=N-4. These are just my best guesses, based on the number of parameters that are estimated in each case.

Anyway, whether you’re comparing two tau-as or two tau-bs, the test statistic is going to be, say: z = (txy – tab)/SEtxy-tab

Getting the SE is the hard part.

The big question is whether the two taus you have are INDEPENDENT or DEPENDENT. If you’re comparing taus between-subjects and neither of the taus draw on the same variable, it’s independent. If you’re comparing taus between-subjects and the two taus both draw on the same variable (i.e., you’ve got txy and txz), then it’s dependent. If you’re comparing taus within-subjects, whether they draw on the same variable or not, then it’s dependent.

SE of Difference in Independent case (Cliff, 1996b, p 341):



Note this is the same approach we’d take for calculating the pooled standard error for the between-subjects t-test (comparing two independent means). Whether you’re comparing tau-as or tau-bs, you just compute the individual SEs as above and then use those to get the pooled SE and you’re all set. Too easy. (Note also that SE^2 is the same as variance here.)

SE of Difference in **Dependent** case (Cliff, 1996b, p 341):



Note this is the same approach we’d take for calculating the pooled standard error for the within-subjects t-test (comparing two dependent means). Actually in those cases we usually take the shortcut and just compute single difference score for each subject and then do a single-sample t-test on that, which saves us the trouble of calculating covariances. Alas, no such shortcut is available when comparing taus, because we only have two overall tau values, not two values for each subject.

We’ve already gone over how to get the variance for tau-a and tau-b. Obviously, the covariance component is the biggest pain in the ass here. If you’re comparing two dependent tau-as, then you can use the same (consistent) formula used earlier to get the covariance and then be on your way. If you’re comparing two dependent tau-bs, you’re basically screwed. No, just kidding, it’s possible, but extremely difficult. This is virtually guaranteed to be most elite thing you’ll have to do all day.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**(VI) CALCULATING THE COVARIANCE BETWEEN TAU-Bs**

(One note first: This REQUIRES equal sample sizes for the data sets for both taus, and that the data be matched/paired across all variables you’re using. So be sure to exclude from analysis any subjects who are missing even a single datum on any of the variables involved.)

Welcome to the rarefied super-elite status of being one of probably just a handful of people in the world (as of 2-4-2010) who are bad-ass enough to attempt this. There’s only ONE place EVER (so far, and that I’ve found) that anyone has figured out and published the way to calculate the covariance between two tau-bs, and (get this) the formula printed has a mistake! Holy crap. Okay, here it is, from Cliff & Charlin (1991, formula 20), brace yourselves foolz!!!:



Ah yes, matrix algebra. And yes, sorry, that’s the best resolution copy I have (from the PDF of the article from the journal; could probably get a nice image from a paper copy).

The first thing to note is that, although the purpose of this formula is to calculate the covariance between two tau-bs, the contents of these matrices are actually all going to be based on the tau-a values. So you’ll either be multiplying together several tau-a values, or computing the covariance between two tau-as.

The second thing to note is that this formula uses subscripts 1,2,3,4 to refer to the 4 different variables involved, and are equivalent to what I’ve been referring to as: x, y, a, b (though keep in mind, in some cases there will really be only 3 variables total involved because you’re comparing taus that draw on a common variable, like say txy vs tay. but the same formula above would still be used… I think. you’d just do, say, cov(txy,tay),cov(txy,taa),cov(txy,tyy) for the top row of the 2nd matrix, and the bottom-right cell of the 2nd matrix would just end up being a variance, that is: cov(tyy,tyy)=var(tyy) ).

There are 3 matrices. (REMEMBER: matrix multiplication is not commutative, so order matters! A\*B≠B\*A necessarily.) The first matrix has one column and three rows. In case you can’t tell, the first item in the first matrix, converting to my notation of 1=x,2=y,3=a,4=b, is: (txx\*tyy)^-.5 . The second item in the 1st matrix is: (-.5)\*(txy)\*(txx^(-3/2))\*(tyy^-.5) . Etc.

**HERE’S THE BIG MISTAKE in the above formula:** The first matrix should be HORIZONTAL, not vertical! That is, it should have 1 row and 3 columns, rather than 3 rows and 1 column. If it’s vertical, then the matrices cannot be multiplied together. So rotate the goddamn thing first. The topmost item should become the leftmost, and the bottommost item should become the rightmost.

The 3rd matrix is just like the 1st matrix except using the second set of variables (a and b in my notation). But the third matrix SHOULD be vertical here.

The 2nd matrix is just awful. It’s got nine different covariances in it, none of which you’ve probably already calculated. The good news is that you can just use the consistent tau-a covariance formulas as provided in section IV above. The bad news is that this is a huge pain in the ass. You’re going to have to go through all that crap of summing squared deviations and everything for a bunch more different combinations of variables. But that all works the same way as it did before.

When you get your matrices filled in, you’ll do matrix1 \* matrix2 = matrixTemp. Then do matrixTemp \* matrix3. Go to <http://en.wikipedia.org/wiki/Matrix_multiplication> for info on how to do matrix multiplication, or just get a computer program to do it for you. This will result in ONE NUMBER! That number is the covariance between tau-b for xy and tau-b for ab. Congratulations. Now you can plug that into the formula in section V for the SE of Difference in Dependent case, and you can finally compute your test statistic. Or make a confidence interval (which I’ve conspicuously not covered here, but should be do-able from all this info, and also Cliff and Woods talk about it, etc.).

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**References**

Cliff, N. (1996a). *Ordinal methods for behavioral data analysis*. Mahwah, NJ: Erlbaum.

Cliff, N. (1996b). Answering ordinal questions with ordinal data using ordinal statistics. *Multivariate Behavioral Research, 31*(3), 331-350.

Cliff, N., & Charlin, V. (1991). Variances and covariances of Kendall’s tau and their estimation. *Multivariate Behavioral Research, 26*, 693–707.

Long, J. D., & Cliff, N. (1997). Confidence intervals for Kendall’s tau. *British Journal of Mathematical and Statistical Psychology, 50*, 31–41.

Woods, C. M. (2007). Confidence intervals for gamma-family measures of ordinal association. *Psychological Methods, 12*, 185–204.

Woods, C. M. (2009). Consistent small-sample variances for six gamma-family measures of ordinal association. *Multivariate Behavioral Research, 44*, 525-551.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

R code from Woods 2009.

<http://www.artsci.wustl.edu/~cwoods/CCvarR.html>

# compute CC variances for all gamma family measures

# Carol M. Woods, 2008

rm(list = ls())

# read in case record data from text file

raw = scan(file="C:/CC variances/empirical example/crs.txt", what=list(row=0,col=0))

n = 30 #number of observations (sum of all table frequencies)

ncols = 3 #number of ordered categories for the column variable

nrows = 3 #number of ordered categories for the row variable

RowVar = raw$row

ColVar = raw$col

# define function (below) before calling it

CCse(n,RowVar,ColVar)

# the function writes results to an output file: GamFamCIs.out

# on my machine, the output file is created here: C:\Program

Files\R\rw2010

####################### function definition ########################

CCse <- function(n, RowVar, ColVar) {

tij = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijxy

t2ij = matrix(c(0),nrow=n,ncol=n) #tau-sub-ijxy SQUARED

tijxx = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijxx, no ties on X

tijyy = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijyy, no ties on Y

sumtidot = matrix(c(0),nrow=1,ncol=n)

sumt2idot = matrix(c(0),nrow=1,ncol=n)

sumtidotxx = matrix(c(0),nrow=1,ncol=n)

sumtidotyy = matrix(c(0),nrow=1,ncol=n)

tidot = matrix(c(0),nrow=1,ncol=n)

t2idot = matrix(c(0),nrow=1,ncol=n)

tidotxx = matrix(c(0),nrow=1,ncol=n)

tidotyy = matrix(c(0),nrow=1,ncol=n)

sumtij=sumt2ij=sumtijxx=sumtijyy=txx=tyy=txy=t2=0

vartidot=vart2idot=vartij=vart2ij=vartijxx=vartijyy=0

covXxXy=vartidotxx=varTxx=covYyXy=vartidotyy=varTyy=0

sti=sti2=stij=st2ij=stixx=stijxx=stiyy=stijyy=0

varTxyC=varT2C=varTxxC=varTyyC=0

varDyxC=varDxyC=varbC=varTauC=varKim=varGam=varE=varLgyx=varLgxy=0

seTxyC=seDyxC=seDxyC=sebC=SeTauC=SeKim=seGam=seLgyx=seE=seLgxy=0

part1c=part2c=0

derivTauc=derivTxyKim=derivTxxKim=derivTyyKim=derivTxyGam=derivT2Gam=0

dTxyLgyx=dTxxLgyx=dT2Lgyx=dTxyLgxy=dTyyLgxy=dT2Lgxy=dTxyE=dTxxE=dTyyE=dT2E=0

ta=tb=sdy=sdx=gamma=wilson\_e=tauc=lgyx=lgxy=kimsym=0

covijxx=covijyy=covij2=covidotxx=covidotyy=covidot2=covijxxC=covijyyC=covij2C=0

covidotxxC=covidotyyC=covidot2C=covXxXyC=covYyXyC=covXxYyC=cov2XyC=covidotxxyy=0

covijxxyy=covidotxxyyC=covijxxyyC=cov2XxC=covidot2XxC=covij2XxC=covidot2xx=0

covij2xx=cov2YyC=covidot2YyC=covij2YyC=covidot2yy=covij2yy=0

outfile = file(description="GamFamCIs.out", open="a")

for (i in 1:n) {

for (j in 1:n) {

if ( ((RowVar[i] > RowVar[j]) && (ColVar[i] > ColVar[j])) || (RowVar[i] < RowVar[j]) && (ColVar[i] < ColVar[j])) {

tij[i,j] = 1

t2ij[i,j] = 1 #square of tij

tijxx[i,j] = 1

tijyy[i,j] = 1

} #end if concordant

if ( ((RowVar[i] > RowVar[j]) && (ColVar[i] < ColVar[j])) || (RowVar[i] < RowVar[j]) && (ColVar[i] > ColVar[j])) {

tij[i,j] = -1

t2ij[i,j] = 1

tijxx[i,j] = 1

tijyy[i,j] = 1

} #end if discordant

if ((RowVar[i] == RowVar[j]) && (ColVar[i] != ColVar[j])) {

tij[i,j] = 0

t2ij[i,j] = 0

tijxx[i,j] = 0

tijyy[i,j] = 1

} #end if ties on X only

if ((RowVar[i] != RowVar[j]) && (ColVar[i] == ColVar[j])) {

tij[i,j] = 0

t2ij[i,j] = 0

tijxx[i,j] = 1

tijyy[i,j] = 0

} #end if ties on Y only

if ((RowVar[i] == RowVar[j]) && (ColVar[i] == ColVar[j])) {

tij[i,j] = 0

t2ij[i,j] = 0

tijxx[i,j] = 0

tijyy[i,j] = 0

} #end if ties on both X and Y

} #end j

} #end i

for (i in 1:n) {

for (j in 1:n) {

sumtij = sumtij + tij[i,j]

sumt2ij = sumt2ij + t2ij[i,j]

sumtijxx = sumtijxx + tijxx[i,j]

sumtijyy = sumtijyy + tijyy[i,j]

sumtidot[i] = sumtidot[i] + tij[i,j]

sumt2idot[i] = sumt2idot[i] + t2ij[i,j]

sumtidotxx[i] = sumtidotxx[i] + tijxx[i,j]

sumtidotyy[i] = sumtidotyy[i] + tijyy[i,j]

}

}

tidot = sumtidot/(n-1) #tidotxy for tau-xy = tau-a

t2idot = sumt2idot/(n-1) #for tijxy squared

tidotxx = sumtidotxx/(n-1) #tau sub i.xx for Somers' dyx

tidotyy = sumtidotyy/(n-1) #for Somers' dxy

txy = sumtij/(n\*(n-1)) #this is tau-a

t2 = sumt2ij/(n\*(n-1)) #(square tijxy then sum over pairs)

txx = sumtijxx/(n\*(n-1))

tyy = sumtijyy/(n\*(n-1))

#(co)variances of tau-ijxy, -ijxx, -ijyy

for (i in 1:n) {

for (j in 1:n) {

if (i!=j) { #important for variances...covariances too

stij = stij + ((tij[i,j] - txy)\*(tij[i,j] - txy))

st2ij = st2ij + ((t2ij[i,j] - t2) \*(t2ij[i,j] - t2))

stijxx = stijxx + ((tijxx[i,j] - txx)\*(tijxx[i,j] - txx)) #for Somers' dyx

stijyy = stijyy + ((tijyy[i,j] - tyy)\*(tijyy[i,j] - tyy)) #for Somers' dxy

covijxx = covijxx + ((tijxx[i,j] - txx)\*(tij[i,j] - txy))

covijyy = covijyy + ((tijyy[i,j] - tyy)\*(tij[i,j] - txy))

covijxxyy = covijxxyy + ((tijyy[i,j] - tyy)\*(tijxx[i,j] - txx)) #for tau-b

covij2 = covij2 + ((t2ij[i,j] - t2) \* (tij[i,j] - txy))

covij2xx = covij2xx + ((t2ij[i,j] - t2) \* (tijxx[i,j] - txx))

covij2yy = covij2yy + ((t2ij[i,j] - t2) \* (tijyy[i,j] - tyy))

}

}

}

vartij = stij/((n\*(n-1))-1)

vart2ij = st2ij/((n\*(n-1))-1)

vartijxx = stijxx/((n\*(n-1))-1)

vartijyy = stijyy/((n\*(n-1))-1)

covijxxC = covijxx/((n\*(n-1))-1)

covijyyC = covijyy/((n\*(n-1))-1)

covijxxyyC = covijxxyy/((n\*(n-1))-1)

covij2C = covij2/((n\*(n-1))-1)

covij2XxC = covij2xx/((n\*(n-1))-1)

covij2YyC = covij2yy/((n\*(n-1))-1)

#(co)variances of tau-i.xy, -i.xx, -i.yy

for (i in 1:n){

sti = sti + ((tidot[i] - txy) \*(tidot[i] - txy))

sti2 = sti2 + ((t2idot[i] - t2) \*(t2idot[i] - t2))

stixx = stixx + ((tidotxx[i] - txx)\*(tidotxx[i] - txx))

#for Somers' dyx

stiyy = stiyy + ((tidotyy[i] - tyy)\*(tidotyy[i] - tyy))

#for Somers' dxy

covidotxx = covidotxx + ((tidotxx[i] - txx)\*(tidot[i] - txy))

covidotyy = covidotyy + ((tidotyy[i] - tyy)\*(tidot[i] - txy))

covidotxxyy = covidotxxyy + ((tidotyy[i] - tyy)\*(tidotxx[i] - txx)) #for tau-b

covidot2 = covidot2 + ((t2idot[i] - t2) \* (tidot[i] - txy))

covidot2xx = covidot2xx + ((t2idot[i] - t2) \* (tidotxx[i] - txx))

covidot2yy = covidot2yy + ((t2idot[i] - t2) \* (tidotyy[i] - tyy))

}

vartidot = sti/(n-1)

vart2idot = sti2/(n-1) #for tijxy squared

vartidotxx = stixx/(n-1)

vartidotyy = stiyy/(n-1)

covidotxxC = covidotxx/(n-1)

covidotyyC = covidotyy/(n-1)

covidotxxyyC = covidotxxyy/(n-1)

covidot2C = covidot2/(n-1)

covidot2XxC = covidot2xx/(n-1)

covidot2YyC = covidot2yy/(n-1)

#variances

#for tau-a = tau-xy

varTxyC = (((4\*(n-2))\*vartidot) + (2\*vartij))/(n\*(n-1))

seTxyC = sqrt(varTxyC) #no division by n

#for tau-xx

varTxxC = (((4\*(n-2))\*vartidotxx) + (2\*vartijxx))/(n\*(n-1))

#for tau-yy

varTyyC = (((4\*(n-2))\*vartidotyy) + (2\*vartijyy))/(n\*(n-1))

#for tau-2 which is tau-xy squared

varT2C = (((4\*(n-2))\*vart2idot) + (2\*vart2ij))/(n\*(n-1))

#covariances

#between tau-xx and tau-xy

covXxXyC = ((4\*(n-2)\*covidotxxC) + (2\*covijxxC))/(n\*(n-1))

#between tau-yy and tau-xy

covYyXyC = ((4\*(n-2)\*covidotyyC) + (2\*covijyyC))/(n\*(n-1))

#between tau-xx and tau-yy

covXxYyC = ((4\*(n-2)\*covidotxxyyC) + (2\*covijxxyyC))/(n\*(n-1))

#between tau-xy and tau-2

cov2XyC = ((4\*(n-2)\*covidot2C) + (2\*covij2C))/(n\*(n-1))

#between tau-xx and tau-2

cov2XxC = ((4\*(n-2)\*covidot2XxC) + (2\*covij2XxC))/(n\*(n-1))

#between tau-yy and tau-2

cov2YyC = ((4\*(n-2)\*covidot2YyC) + (2\*covij2YyC))/(n\*(n-1))

## CC variances ##

#tau-c

m = ncols #ncols and nrows are global variables

if (nrows < ncols) m = nrows

derivTauc = (n\*m\*(n-1))/((n\*n)\*(m-1))

varTauC = derivTauc\*derivTauc\*varTxyC

SeTauC = sqrt(varTauC)

#Kim's symmetric d

derivTxyKim = 2/(tyy+txx)

derivTxxKim = (-2\*txy)/((tyy+txx)\*(tyy+txx))

derivTyyKim = derivTxxKim

varKim = (derivTxyKim\*derivTxyKim\*varTxyC) + (derivTxxKim\*derivTxxKim\*varTxxC) + (derivTyyKim\*derivTyyKim\*varTyyC) + (2\*derivTxyKim\*derivTyyKim\*covYyXyC) +

(2\*derivTxyKim\*derivTxxKim\*covXxXyC) + (2\*derivTxxKim\*derivTyyKim\*covXxYyC)

SeKim = sqrt(varKim)

#gamma

derivTxyGam = 1/t2

derivT2Gam = -txy/(t2\*t2)

varGam = (derivTxyGam\*derivTxyGam\*varTxyC) + (derivT2Gam\*derivT2Gam\*varT2C) + (2\*derivTxyGam\*derivT2Gam\*cov2XyC)

seGam = sqrt(varGam)

#Leik-Gove d-Yx

dTxyLgyx = 1/((2\*txx)- t2) #deriv w.r.t. txy

dTxxLgyx = (-2\*txy)/(((2\*txx)-t2)\*((2\*txx)-t2)) #deriv w.r.t. txx

dT2Lgyx = txy/(((2\*txx)-t2)\*((2\*txx)-t2)) #deriv w.r.t. t2

varLgyx = (dTxyLgyx\*dTxyLgyx\*varTxyC) + (dTxxLgyx\*dTxxLgyx\*varTxxC) + (dT2Lgyx\*dT2Lgyx\*varT2C) + (2\*dTxyLgyx\*dTxxLgyx\*covXxXyC) +

(2\*dTxyLgyx\*dT2Lgyx\*cov2XyC) + (2\*dTxxLgyx\*dT2Lgyx\*cov2XxC)

seLgyx = sqrt(varLgyx)

#Leik-Gove d-Xy

dTxyLgxy = 1/((2\*tyy)-t2) #deriv wrt txy

dTyyLgxy = (-2\*txy)/(((2\*tyy)-t2)\*((2\*tyy)-t2)) #deriv wrt tyy

dT2Lgxy = txy/(((2\*tyy)-t2)\*((2\*tyy)-t2)) #deriv wrt t2

varLgxy = (dTxyLgxy\*dTxyLgxy\*varTxyC) + (dTyyLgxy\*dTyyLgxy\*varTyyC) + (dT2Lgxy\*dT2Lgxy\*varT2C) + (2\*dTxyLgxy\*dTyyLgxy\*covYyXyC) +

(2\*dTxyLgxy\*dT2Lgxy\*cov2XyC) + (2\*dTyyLgxy\*dT2Lgxy\*cov2YyC)

seLgxy = sqrt(varLgxy)

#Wilson e

dTxyE = 1/(txx + tyy - t2) #deriv wrt txy

dTxxE = -txy/((txx + tyy - t2)\*(txx + tyy - t2)) #deriv wrt txx

dTyyE = dTxxE #deriv wrt tyy is the same

dT2E = txy/((txx + tyy - t2)\*(txx + tyy - t2)) #deriv wrt t2 numerator positive

varE = (dTxyE\*dTxyE\*varTxyC) + (dTxxE\*dTxxE\*varTxxC) + (dTyyE\*dTyyE\*varTyyC) + (dT2E\*dT2E\*varT2C) + (2\*dTxyE\*dTxxE\*covXxXyC) + (2\*dTxyE\*dTyyE\*covYyXyC) +

(2\*dTxyE\*dT2E\*cov2XyC) + (2\*dTxxE\*dTyyE\*covXxYyC) + (2\*dTxxE\*dT2E\*cov2XxC) + (2\*dTyyE\*dT2E\*cov2YyC)

seE = sqrt(varE)

#tau-b

#(from Cliff 1996 p.79 but includes denom. from Cliff & Charlin p. 700)

part1c = varTxyC - (txy\*((covXxXyC/txx)+(covYyXyC/tyy)))

part2c = (txy\*txy)\*((varTxxC/(4\*txx\*txx))+(varTyyC/(4\*tyy\*tyy))+(covXxYyC/(2\*txx\*tyy)))

varbC = (part1c + part2c)/(txx\*tyy)

sebC = sqrt(varbC)

#Somers' dyx

varDyxC = (varTxyC/(txx\*txx)) - (((2\*txy)\*covXxXyC)/(txx\*txx\*txx)) + ( (txy\*txy\*varTxxC)/(txx\*txx\*txx\*txx))

seDyxC = sqrt(varDyxC)

#Somers' dxy varDxyC = (varTxyC/(tyy\*tyy)) - (((2\*txy)\*covYyXyC)/(tyy\*tyy\*tyy)) + ( (txy\*txy\*varTyyC)/(tyy\*tyy\*tyy\*tyy))

seDxyC = sqrt(varDxyC)

#point estimates of gamma family indices

ta = txy

tb = txy/(sqrt(txx\*tyy))

gamma = txy/t2

wilson\_e = sumtij/(sumtijxx + sumtijyy - sumt2ij)

m = ncols #ncols and nrows are global

if (nrows < ncols) m = nrows

tauc = sumtij/((n\*n)\*((m-1)/m))

sdy = txy/txx

sdx = txy/tyy

lgyx = sumtij/((2\*sumtijxx) - sumt2ij)

lgxy = sumtij/((2\*sumtijyy) - sumt2ij)

kimsym = sumtij/(.5\*(sumtijyy + sumtijxx))

#to output file

cat(

" gamma: ", gamma, " SE: ", seGam, "\n", "\n",

"tau-a: ", ta, " SE: ", seTxyC, "\n", "\n",

"tau-b: ", tb, " SE: ", sebC, "\n", "\n",

"wilson e: ", wilson\_e, " SE: ", seE, "\n", "\n",

"tau-c: ", tauc, " SE: ", SeTauC, "\n", "\n",

"Somers' dyx: ", sdy, " SE: ", seDyxC, "\n", "\n",

"Somers' dxy: ", sdx, " SE: ", seDxyC, "\n", "\n",

"leik-gove yx: ", lgyx, " SE: ", seLgyx, "\n", "\n",

"leik-gove xy: ", lgxy, " SE: ", seLgxy, "\n", "\n",

"kims symmetric: ", kimsym, " SE: ", SeKim, "\n",

file=outfile,append=TRUE)

close(outfile)

} #end CCse

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Some R code I adapted from Woods 2009, gives results for comparing two tau-bs for independent and dependent case.

####################### function definition; adapted by Jason R Finley 2-4-2010 from Woods 2009 ########################

CCseJRFc <- function(X, Y, A, B, outputname) {

#there MUST be the same number of subjects in all 4 columns submitted to this function.

#be sure to remove rows that have any empty cells before calling this function.

n = length(X)

####### Declare variables for X and Y

tijxy = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijxy

t2ijxy = matrix(c(0),nrow=n,ncol=n) #tau-sub-ijxy SQUARED

tijxx = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijxx, no ties on X

tijyy = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijyy, no ties on Y

sumtidotxy = matrix(c(0),nrow=1,ncol=n)

sumt2idotxy = matrix(c(0),nrow=1,ncol=n)

sumtidotxx = matrix(c(0),nrow=1,ncol=n)

sumtidotyy = matrix(c(0),nrow=1,ncol=n)

tidotxy = matrix(c(0),nrow=1,ncol=n)

t2idotxy = matrix(c(0),nrow=1,ncol=n)

tidotxx = matrix(c(0),nrow=1,ncol=n)

tidotyy = matrix(c(0),nrow=1,ncol=n)

sumtijxy=sumt2ijxy=sumtijxx=sumtijyy=txx=tyy=txy=t2xy=0

vartidotxy=vart2idotxy=vartijxy=vart2ijxy=vartijxx=vartijyy=0

covXxXy=vartidotxx=varTxx=covYyXy=vartidotyy=varTyy=0

stixy=sti2xy=stijxy=st2ijxy=stixx=stijxx=stiyy=stijyy=0

varTxyC=varT2xyC=varTxxC=varTyyC=0

varbxyC=varGamxy=0

sebxyC=seGamxy=0

part1cxy=part2cxy=0

derivTxyGam=derivT2xyGam=0

txya=txyb=gammaxy=0

covijXxXy=covijYyXy=covij2XyXy=covidotXxXy=covidotYyXy=covidot2XyXy=covijXxXyC=covijYyXyC=covij2XyXyC=0

covidotXxXyC=covidotYyXyC=covidot2XyXyC=covXxXyC=covYyXyC=covXxYyC=cov2XyXyC=covidotXxYy=0

covijXxYy=covidotXxYyC=covijXxYyC=cov2XxXyC=covidot2XxXyC=covij2XxXyC=covidot2XxXy=0

covij2XxXy=cov2YyXyC=covidot2YyXyC=covij2YyXyC=covidot2YyXy=covij2YyXy=0

#####################

####### Declare variables for A and B

tijab = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijab

t2ijab = matrix(c(0),nrow=n,ncol=n) #tau-sub-ijab SQUARED

tijaa = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijaa, no ties on X

tijbb = matrix(c(0),nrow=n,ncol=n) #this is tau-sub-ijbb, no ties on Y

sumtidotab = matrix(c(0),nrow=1,ncol=n)

sumt2idotab = matrix(c(0),nrow=1,ncol=n)

sumtidotaa = matrix(c(0),nrow=1,ncol=n)

sumtidotbb = matrix(c(0),nrow=1,ncol=n)

tidotab = matrix(c(0),nrow=1,ncol=n)

t2idotab = matrix(c(0),nrow=1,ncol=n)

tidotaa = matrix(c(0),nrow=1,ncol=n)

tidotbb = matrix(c(0),nrow=1,ncol=n)

sumtijab=sumt2ijab=sumtijaa=sumtijbb=taa=tbb=tab=t2ab=0

vartidotab=vart2idotab=vartijab=vart2ijab=vartijaa=vartijbb=0

covAaAb=vartidotaa=varTaa=covBbAb=vartidotbb=varTbb=0

stiab=sti2ab=stijab=st2ijab=stiaa=stijaa=stibb=stijbb=0

varTabC=varT2abC=varTaaC=varTbbC=0

varbabC=varGamab=0

sebabC=seGamab=0

part1cab=part2cab=0

derivTabGam=derivT2abGam=0

taba=tabb=gammaab=0

covijAaAb=covijBbAb=covij2AbAb=covidotAaAb=covidotBbAb=covidot2AbAb=covijAaAbC=covijBbAbC=covij2AbAbC=0

covidotAaAbC=covidotBbAbC=covidot2AbAbC=covAaAbC=covBbAbC=covAaBbC=cov2AbAbC=covidotAaBb=0

covijAaBb=covidotAaBbC=covijAaBbC=cov2AaAbC=covidot2AaAbC=covij2AaAbC=covidot2AaAb=0

covij2AaAb=cov2BbAbC=covidot2BbAbC=covij2BbAbC=covidot2BbAb=covij2BbAb=0

#####################

#declare variables for covariances of X/Y and A/B

covijXyAb = covijXyAa = covijXyBb = covijXxAb = covijXxAa = covijXxBb = covijYyAb = covijYyAa = covijYyBb = 0

covijXyAbC = covijXyAaC = covijXyBbC = covijXxAbC = covijXxAaC = covijXxBbC = covijYyAbC = covijYyAaC = covijYyBbC = 0

covidotXyAb = covidotXyAa = covidotXyBb = covidotXxAb = covidotXxAa = covidotXxBb = covidotYyAb = covidotYyAa = covidotYyBb = 0

covidotXyAbC = covidotXyAaC = covidotXyBbC = covidotXxAbC = covidotXxAaC = covidotXxBbC = covidotYyAbC = covidotYyAaC = covidotYyBbC = 0

covXyAbC = covXyAaC = covXyBbC = covXxAbC = covXxAaC = covXxBbC = covYyAbC = covYyAaC = covYyBbC = 0

### Loop through all possible pairings for X & Y and count up concordances, discordances, ties

for (i in 1:n) {

for (j in 1:n) {

#each ij pairing is only going to satisfy one of the following 5 IF statements... (I think)

if ( ((X[i] > X[j]) && (Y[i] > Y[j])) || (X[i] < X[j]) && (Y[i] < Y[j])) {

tijxy[i,j] = 1 #aka tij

t2ijxy[i,j] = 1 #square of tij

tijxx[i,j] = 1

tijyy[i,j] = 1

} #end if concordant

if ( ((X[i] > X[j]) && (Y[i] < Y[j])) || (X[i] < X[j]) && (Y[i] > Y[j])) {

tijxy[i,j] = -1

t2ijxy[i,j] = 1

tijxx[i,j] = 1

tijyy[i,j] = 1

} #end if discordant

if ((X[i] == X[j]) && (Y[i] != Y[j])) {

tijxy[i,j] = 0

t2ijxy[i,j] = 0

tijxx[i,j] = 0

tijyy[i,j] = 1

} #end if ties on X only

if ((X[i] != X[j]) && (Y[i] == Y[j])) {

tijxy[i,j] = 0

t2ijxy[i,j] = 0

tijxx[i,j] = 1

tijyy[i,j] = 0

} #end if ties on Y only

if ((X[i] == X[j]) && (Y[i] == Y[j])) {

tijxy[i,j] = 0

t2ijxy[i,j] = 0

tijxx[i,j] = 0

tijyy[i,j] = 0

} #end if ties on both X and Y

} #end j

} #end i

### Loop through all possible pairings for A & B and count up concordances, discordances, ties

for (i in 1:n) {

for (j in 1:n) {

#each ij pairing is only going to satisfy one of the following 5 IF statements... (I think)

if ( ((A[i] > A[j]) && (B[i] > B[j])) || (A[i] < A[j]) && (B[i] < B[j])) {

tijab[i,j] = 1 #aka tij

t2ijab[i,j] = 1 #square of tij

tijaa[i,j] = 1

tijbb[i,j] = 1

} #end if concordant

if ( ((A[i] > A[j]) && (B[i] < B[j])) || (A[i] < A[j]) && (B[i] > B[j])) {

tijab[i,j] = -1

t2ijab[i,j] = 1

tijaa[i,j] = 1

tijbb[i,j] = 1

} #end if discordant

if ((A[i] == A[j]) && (B[i] != B[j])) {

tijab[i,j] = 0

t2ijab[i,j] = 0

tijaa[i,j] = 0

tijbb[i,j] = 1

} #end if ties on A only

if ((A[i] != A[j]) && (B[i] == B[j])) {

tijab[i,j] = 0

t2ijab[i,j] = 0

tijaa[i,j] = 1

tijbb[i,j] = 0

} #end if ties on B only

if ((A[i] == A[j]) && (B[i] == B[j])) {

tijab[i,j] = 0

t2ijab[i,j] = 0

tijaa[i,j] = 0

tijbb[i,j] = 0

} #end if ties on both A and B

} #end j

} #end i

for (i in 1:n) {

for (j in 1:n) {

sumtijxy = sumtijxy + tijxy[i,j]

sumt2ijxy = sumt2ijxy + t2ijxy[i,j]

sumtijxx = sumtijxx + tijxx[i,j]

sumtijyy = sumtijyy + tijyy[i,j]

sumtidotxy[i] = sumtidotxy[i] + tijxy[i,j]

sumt2idotxy[i] = sumt2idotxy[i] + t2ijxy[i,j]

sumtidotxx[i] = sumtidotxx[i] + tijxx[i,j]

sumtidotyy[i] = sumtidotyy[i] + tijyy[i,j]

sumtijab = sumtijab + tijab[i,j]

sumt2ijab = sumt2ijab + t2ijab[i,j]

sumtijaa = sumtijaa + tijaa[i,j]

sumtijbb = sumtijbb + tijbb[i,j]

sumtidotab[i] = sumtidotab[i] + tijab[i,j]

sumt2idotab[i] = sumt2idotab[i] + t2ijab[i,j]

sumtidotaa[i] = sumtidotaa[i] + tijaa[i,j]

sumtidotbb[i] = sumtidotbb[i] + tijbb[i,j]

}

}

tidotxy = sumtidotxy/(n-1) #tidotxy for tau-xy = tau-a

t2idotxy = sumt2idotxy/(n-1) #for tijxy squared

tidotxx = sumtidotxx/(n-1) #tau sub i.xx for Somers' dyx

tidotyy = sumtidotyy/(n-1) #for Somers' dxy

txy = sumtijxy/(n\*(n-1)) #this is tau-a

t2xy = sumt2ijxy/(n\*(n-1)) #(square tijxy then sum over pairs)

txx = sumtijxx/(n\*(n-1))

tyy = sumtijyy/(n\*(n-1))

tidotab = sumtidotab/(n-1) #tidotab for tau-ab = tau-a

t2idotab = sumt2idotab/(n-1) #for tijab squared

tidotaa = sumtidotaa/(n-1) #tau sub i.aa for Somers' dba

tidotbb = sumtidotbb/(n-1) #for Somers' dab

tab = sumtijab/(n\*(n-1)) #this is tau-a

t2ab = sumt2ijab/(n\*(n-1)) #(square tijab then sum over pairs)

taa = sumtijaa/(n\*(n-1))

tbb = sumtijbb/(n\*(n-1))

#HERE1 covij

#(co)variances of tau-ijxy, -ijxx, -ijyy

for (i in 1:n) {

for (j in 1:n) {

if (i!=j) { #important for variances...covariances too

stijxy = stijxy + ((tijxy[i,j] - txy)\*(tijxy[i,j] - txy))

st2ijxy = st2ijxy + ((t2ijxy[i,j] - t2xy) \*(t2ijxy[i,j] - t2xy))

stijxx = stijxx + ((tijxx[i,j] - txx)\*(tijxx[i,j] - txx)) #for Somers' dyx

stijyy = stijyy + ((tijyy[i,j] - tyy)\*(tijyy[i,j] - tyy)) #for Somers' dxy

covijXxXy = covijXxXy + ((tijxx[i,j] - txx)\*(tijxy[i,j] - txy))

covijYyXy = covijYyXy + ((tijyy[i,j] - tyy)\*(tijxy[i,j] - txy))

covijXxYy = covijXxYy + ((tijyy[i,j] - tyy)\*(tijxx[i,j] - txx)) #for tau-b

stijab = stijab + ((tijab[i,j] - tab)\*(tijab[i,j] - tab))

st2ijab = st2ijab + ((t2ijab[i,j] - t2ab) \*(t2ijab[i,j] - t2ab))

stijaa = stijaa + ((tijaa[i,j] - taa)\*(tijaa[i,j] - taa)) #for Somers' dyx

stijbb = stijbb + ((tijbb[i,j] - tbb)\*(tijbb[i,j] - tbb)) #for Somers' dab

covijAaAb = covijAaAb + ((tijaa[i,j] - taa)\*(tijab[i,j] - tab))

covijBbAb = covijBbAb + ((tijbb[i,j] - tbb)\*(tijab[i,j] - tab))

covijAaBb = covijAaBb + ((tijbb[i,j] - tbb)\*(tijaa[i,j] - taa)) #for tau-b

#components needed for covariances between X/Y and A/B

covijXyAb = covijXyAb + ((tijxy[i,j] - txy)\*(tijab[i,j] - tab))

covijXyAa = covijXyAa + ((tijxy[i,j] - txy)\*(tijaa[i,j] - taa))

covijXyBb = covijXyBb + ((tijxy[i,j] - txy)\*(tijbb[i,j] - tbb))

covijXxAb = covijXxAb + ((tijxx[i,j] - txx)\*(tijab[i,j] - tab))

covijXxAa = covijXxAa + ((tijxx[i,j] - txx)\*(tijaa[i,j] - taa))

covijXxBb = covijXxBb + ((tijxx[i,j] - txx)\*(tijbb[i,j] - tbb))

covijYyAb = covijYyAb + ((tijyy[i,j] - tyy)\*(tijab[i,j] - tab))

covijYyAa = covijYyAa + ((tijyy[i,j] - tyy)\*(tijaa[i,j] - taa))

covijYyBb = covijYyBb + ((tijyy[i,j] - tyy)\*(tijbb[i,j] - tbb))

}

}

}

vartijxy = stijxy/((n\*(n-1))-1)

vart2ijxy = st2ijxy/((n\*(n-1))-1)

vartijxx = stijxx/((n\*(n-1))-1)

vartijyy = stijyy/((n\*(n-1))-1)

covijXxXyC = covijXxXy/((n\*(n-1))-1)

covijYyXyC = covijYyXy/((n\*(n-1))-1)

covijXxYyC = covijXxYy/((n\*(n-1))-1)

vartijab = stijab/((n\*(n-1))-1)

vart2ijab = st2ijab/((n\*(n-1))-1)

vartijaa = stijaa/((n\*(n-1))-1)

vartijbb = stijbb/((n\*(n-1))-1)

covijAaAbC = covijAaAb/((n\*(n-1))-1)

covijBbAbC = covijBbAb/((n\*(n-1))-1)

covijAaBbC = covijAaBb/((n\*(n-1))-1)

#components needed for covariances between X/Y and A/B

covijXyAbC = covijXyAb/((n\*(n-1))-1)

covijXyAaC = covijXyAa/((n\*(n-1))-1)

covijXyBbC = covijXyBb/((n\*(n-1))-1)

covijXxAbC = covijXxAb/((n\*(n-1))-1)

covijXxAaC = covijXxAa/((n\*(n-1))-1)

covijXxBbC = covijXxBb/((n\*(n-1))-1)

covijYyAbC = covijYyAb/((n\*(n-1))-1)

covijYyAaC = covijYyAa/((n\*(n-1))-1)

covijYyBbC = covijYyBb/((n\*(n-1))-1)

#HERE2 covidot

#(co)variances of tau-i.xy, -i.xx, -i.yy

for (i in 1:n){

stixy = stixy + ((tidotxy[i] - txy) \*(tidotxy[i] - txy))

sti2xy = sti2xy + ((t2idotxy[i] - t2xy) \*(t2idotxy[i] - t2xy))

stixx = stixx + ((tidotxx[i] - txx)\*(tidotxx[i] - txx))

#for Somers' dyx

stiyy = stiyy + ((tidotyy[i] - tyy)\*(tidotyy[i] - tyy))

#for Somers' dxy

covidotXxXy = covidotXxXy + ((tidotxx[i] - txx)\*(tidotxy[i] - txy))

covidotYyXy = covidotYyXy + ((tidotyy[i] - tyy)\*(tidotxy[i] - txy))

covidotXxYy = covidotXxYy + ((tidotyy[i] - tyy)\*(tidotxx[i] - txx)) #for tau-b

stiab = stiab + ((tidotab[i] - tab) \*(tidotab[i] - tab))

sti2ab = sti2ab + ((t2idotab[i] - t2ab) \*(t2idotab[i] - t2ab))

stiaa = stiaa + ((tidotaa[i] - taa)\*(tidotaa[i] - taa))

#for Somers' dba

stibb = stibb + ((tidotbb[i] - tbb)\*(tidotbb[i] - tbb))

#for Somers' dab

covidotAaAb = covidotAaAb + ((tidotaa[i] - taa)\*(tidotab[i] - tab))

covidotBbAb = covidotBbAb + ((tidotbb[i] - tbb)\*(tidotab[i] - tab))

covidotAaBb = covidotAaBb + ((tidotbb[i] - tbb)\*(tidotaa[i] - taa)) #for tau-b

#components needed for covariances between X/Y and A/B

covidotXyAb = covidotXyAb + ((tidotxy[i] - txy)\*(tidotab[i] - tab))

covidotXyAa = covidotXyAa + ((tidotxy[i] - txy)\*(tidotaa[i] - taa))

covidotXyBb = covidotXyBb + ((tidotxy[i] - txy)\*(tidotbb[i] - tbb))

covidotXxAb = covidotXxAb + ((tidotxx[i] - txx)\*(tidotab[i] - tab))

covidotXxAa = covidotXxAa + ((tidotxx[i] - txx)\*(tidotaa[i] - taa))

covidotXxBb = covidotXxBb + ((tidotxx[i] - txx)\*(tidotbb[i] - tbb))

covidotYyAb = covidotYyAb + ((tidotyy[i] - tyy)\*(tidotab[i] - tab))

covidotYyAa = covidotYyAa + ((tidotyy[i] - tyy)\*(tidotaa[i] - taa))

covidotYyBb = covidotYyBb + ((tidotyy[i] - tyy)\*(tidotbb[i] - tbb))

}

vartidotxy = stixy/(n-1)

vart2idotxy = sti2xy/(n-1) #for tijxy squared

vartidotxx = stixx/(n-1)

vartidotyy = stiyy/(n-1)

covidotXxXyC = covidotXxXy/(n-1)

covidotYyXyC = covidotYyXy/(n-1)

covidotXxYyC = covidotXxYy/(n-1)

vartidotab = stiab/(n-1)

vart2idotab = sti2ab/(n-1) #for tijab squared

vartidotaa = stiaa/(n-1)

vartidotbb = stibb/(n-1)

covidotAaAbC = covidotAaAb/(n-1)

covidotBbAbC = covidotBbAb/(n-1)

covidotAaBbC = covidotAaBb/(n-1)

#components needed for covariances between X/Y and A/B

covidotXyAbC = covidotXyAb/(n-1)

covidotXyAaC = covidotXyAa/(n-1)

covidotXyBbC = covidotXyBb/(n-1)

covidotXxAbC = covidotXxAb/(n-1)

covidotXxAaC = covidotXxAa/(n-1)

covidotXxBbC = covidotXxBb/(n-1)

covidotYyAbC = covidotYyAb/(n-1)

covidotYyAaC = covidotYyAa/(n-1)

covidotYyBbC = covidotYyBb/(n-1)

#CONSISTENT variances

#for tau-a = tau-xy

varTxyC = (((4\*(n-2))\*vartidotxy) + (2\*vartijxy))/(n\*(n-1))

seTxyC = sqrt(varTxyC) #no division by n

#for tau-xx

varTxxC = (((4\*(n-2))\*vartidotxx) + (2\*vartijxx))/(n\*(n-1))

#for tau-yy

varTyyC = (((4\*(n-2))\*vartidotyy) + (2\*vartijyy))/(n\*(n-1))

#for tau-2 which is tau-xy squared

varT2xyC = (((4\*(n-2))\*vart2idotxy) + (2\*vart2ijxy))/(n\*(n-1))

#for tau-a = tau-ab

varTabC = (((4\*(n-2))\*vartidotab) + (2\*vartijab))/(n\*(n-1))

seTabC = sqrt(varTabC) #no division by n

#for tau-aa

varTaaC = (((4\*(n-2))\*vartidotaa) + (2\*vartijaa))/(n\*(n-1))

#for tau-bb

varTbbC = (((4\*(n-2))\*vartidotbb) + (2\*vartijbb))/(n\*(n-1))

#for tau-2 which is tau-ab squared

varT2abC = (((4\*(n-2))\*vart2idotab) + (2\*vart2ijab))/(n\*(n-1))

#CONSISTENT covariances

#between tau-xx and tau-xy

covXxXyC = ((4\*(n-2)\*covidotXxXyC) + (2\*covijXxXyC))/(n\*(n-1))

#between tau-yy and tau-xy

covYyXyC = ((4\*(n-2)\*covidotYyXyC) + (2\*covijYyXyC))/(n\*(n-1))

#between tau-xx and tau-yy

covXxYyC = ((4\*(n-2)\*covidotXxYyC) + (2\*covijXxYyC))/(n\*(n-1))

#between tau-aa and tau-ab

covAaAbC = ((4\*(n-2)\*covidotAaAbC) + (2\*covijAaAbC))/(n\*(n-1))

#between tau-bb and tau-ab

covBbAbC = ((4\*(n-2)\*covidotBbAbC) + (2\*covijBbAbC))/(n\*(n-1))

#between tau-aa and tau-bb

covAaBbC = ((4\*(n-2)\*covidotAaBbC) + (2\*covijAaBbC))/(n\*(n-1))

#9 new covariances

covXyAbC = ((4\*(n-2)\*covidotXyAbC) + (2\*covijXyAbC))/(n\*(n-1))

covXyAaC = ((4\*(n-2)\*covidotXyAaC) + (2\*covijXyAaC))/(n\*(n-1))

covXyBbC = ((4\*(n-2)\*covidotXyBbC) + (2\*covijXyBbC))/(n\*(n-1))

covXxAbC = ((4\*(n-2)\*covidotXxAbC) + (2\*covijXxAbC))/(n\*(n-1))

covXxAaC = ((4\*(n-2)\*covidotXxAaC) + (2\*covijXxAaC))/(n\*(n-1))

covXxBbC = ((4\*(n-2)\*covidotXxBbC) + (2\*covijXxBbC))/(n\*(n-1))

covYyAbC = ((4\*(n-2)\*covidotYyAbC) + (2\*covijYyAbC))/(n\*(n-1))

covYyAaC = ((4\*(n-2)\*covidotYyAaC) + (2\*covijYyAaC))/(n\*(n-1))

covYyBbC = ((4\*(n-2)\*covidotYyBbC) + (2\*covijYyBbC))/(n\*(n-1))

## CC variances ##

#gamma X Y

derivTxyGam = 1/t2xy

derivT2xyGam = -txy/(t2xy\*t2xy)

varGamxy = (derivTxyGam\*derivTxyGam\*varTxyC) + (derivT2xyGam\*derivT2xyGam\*varT2xyC) + (2\*derivTxyGam\*derivT2xyGam\*cov2XyXyC)

seGamxy = sqrt(varGamxy)

#tau-b X Y

#(from Cliff 1996 p.79 but includes denom. from Cliff & Charlin p. 700)

part1cxy = varTxyC - (txy\*((covXxXyC/txx)+(covYyXyC/tyy)))

part2cxy = (txy\*txy)\*((varTxxC/(4\*txx\*txx))+(varTyyC/(4\*tyy\*tyy))+(covXxYyC/(2\*txx\*tyy)))

varbxyC = (part1cxy + part2cxy)/(txx\*tyy)

sebxyC = sqrt(varbxyC)

#point estimates of gamma family indices X Y

taxy = txy

tbxy = txy/(sqrt(txx\*tyy))

gammaxy = txy/t2xy

#gamma A B

derivTabGam = 1/t2ab

derivT2abGam = -tab/(t2ab\*t2ab)

varGamab = (derivTabGam\*derivTabGam\*varTabC) + (derivT2abGam\*derivT2abGam\*varT2abC) + (2\*derivTabGam\*derivT2abGam\*cov2AbAbC)

seGamab = sqrt(varGamab)

#tau-b A B

#(from Cliff 1996 p.79 but includes denom. from Cliff & Charlin p. 700)

part1cab = varTabC - (tab\*((covAaAbC/taa)+(covBbAbC/tbb)))

part2cab = (tab\*tab)\*((varTaaC/(4\*taa\*taa))+(varTbbC/(4\*tbb\*tbb))+(covAaBbC/(2\*taa\*tbb)))

varbabC = (part1cab + part2cab)/(taa\*tbb)

sebabC = sqrt(varbabC)

#point estimates of gamma family indices A B

taab = tab

tbab = tab/(sqrt(taa\*tbb))

gammaab = tab/t2ab

#NOWWWW!!!! Get the covariance between the two tau-b coefficients!!!!

#from Cliff & Charlin 1991, formula 20

mat1.1=(txx\*tyy)^(-.5)

mat1.2=(-.5)\*(txy)\*(txx^(-(3/2)))\*(tyy^(-.5))

mat1.3=(-.5)\*(txy)\*(txx^(-.5))\*(tyy^(-(3/2)))

mat1 <- matrix(c(mat1.1,mat1.2,mat1.3),1,3)

mat2 <- matrix(c(covXyAbC, covXyAaC, covXyBbC, covXxAbC, covXxAaC, covXxBbC, covYyAbC, covYyAaC, covYyBbC),3,3)

mat3.1=(taa\*tbb)^(-.5)

mat3.2=(-.5)\*(tab)\*(taa^(-(3/2)))\*(tbb^(-.5))

mat3.3=(-.5)\*(tab)\*(taa^(-.5))\*(tbb^(-(3/2)))

mat3 <- matrix(c(mat3.1,mat3.2,mat3.3),3,1)

covarTaubXYAB <- mat1 %\*% mat2 %\*% mat3

seTaubXYABind <- sqrt((sebxyC^2) + (sebabC^2))

seTaubXYABdep <- sqrt(varbxyC + varbabC - (2\*covarTaubXYAB))

zInd <- (tbxy-tbab)/seTaubXYABind

zDep <- (tbxy-tbab)/seTaubXYABdep

#Cliff 1996b, p 341, says to use z-test to compare taus. (other main option would be t-test, but I'm not sure what appropriate df would be. maybe: for ind: N1+N2-4. for dep with two taus drawing on same variable: N-3. for two dep with two taus drawing on different variables: N-4.

pInd<-2\*(1-pnorm(abs(zInd)))

pDep<-2\*(1-pnorm(abs(zDep)))

outfile = file(description=paste(outputname,"\_CompareTaub.txt",sep=""), open="a") #on Mac OSX saves to home directory of current user.

#to output file

cat(

" gamma XY: ", gammaxy, " SE: ", seGamxy, "\n",

"tau-a XY: ", taxy, " SE: ", seTxyC, "\n",

"tau-b XY: ", tbxy, " SE: ", sebxyC, "\n",

"gamma AB: ", gammaab, " SE: ", seGamab, "\n",

"tau-a AB: ", taab, " SE: ", seTabC, "\n",

"tau-b AB: ", tbab, " SE: ", sebabC, "\n\n",

"SE (tau-b(XY) - tau-b(AB)) if independent: ", seTaubXYABind, "\n",

"SE (tau-b(XY) - tau-b(AB)) if dependent: ", seTaubXYABdep, "\n",

"indepdenent Z: ",zInd, " p: ",pInd,"\n",

"dependent Z: ",zDep, " p: ",pDep,"\n",

file=outfile,append=TRUE)

close(outfile)

} #end