**Matrix**

|  |  |  |
| --- | --- | --- |
| **Notes** |  |  |
| Output Created |  | 09-JUN-2024 12:41:45 |
| Comments |  |  |
| Input | Active Dataset | DataSet5 |
| Filter | <none> |
| Weight | <none> |
| Split File | <none> |
| N of Rows in Working Data File | 1080 |
| Syntax |  | MATRIX. get dat/variables = Coder1 Coder2 /file = \*/names = vn/missing = -9999999. compute btn = 10000. do if ( 10000 > 0). compute btn = trunc( 10000 /1000)\*1000. end if. do if ( 10000 > 0 and btn = 0). print/title = 'Number of bootstraps must be at least 1000.'. end if. compute btprob = 0. compute rw = 1. loop i = 1 to nrow(dat). compute good = 0. loop j = 1 to ncol(dat). do if (dat(i,j) <> -9999999). compute good = good + 1. end if. end loop. do if (good > 1). compute dat(rw,:) = dat(i,:). compute rw = rw+1. end if. end loop. compute dat = dat(1:(rw-1),:). compute nj = ncol(dat). compute nobj = nrow(dat). compute dat3 = dat. compute m = reshape(t(dat),(nobj\*nj),1). compute allm = nobj\*nj. compute j = 0. loop i = 1 to nrow(m). do if m(i,1) <> -9999999. compute j = j + 1. compute m(j,:)=m(i,:). end if. end loop. compute m = m(1:j,1). compute mss = nrow(m). compute mss = allm-mss. compute mtmp = m. compute mtmp(GRADE(m)) = m. compute m = mtmp. compute m2 = make(nrow(m),1,m(1,1)). compute yass = csum((m = m2))/nrow(m). do if (yass <> 1). compute des = design(m). compute uniq = ncol(des). compute coinc = make(uniq,uniq,0). compute delta = coinc. compute map = make(uniq,1,0). loop i = 1 to nrow(m). loop j = 1 to uniq. do if (des(i,j) = 1). compute map(j,1) = m(i,1). end if. end loop. end loop. loop i = 1 to nobj. loop j = 1 to nj. do if dat(i,j) <> -9999999. loop k = 1 to uniq. do if dat(i,j) = map(k,1). compute dat(i,j) = k. BREAK. end if. end loop. end if. end loop. end loop. compute datms = (dat <> -9999999). compute mu = rsum(datms). compute nprs = csum(mu&\*(mu-1))\*.5. compute btalp = make((btn+1),1,-999). loop k = 1 to nobj. compute temp = make(uniq, uniq, 0). loop i = 1 to nj. loop j = 1 to nj. do if (dat(k,i) <> -9999999 AND dat(k,j) <> -9999999 AND i <> j). compute temp(dat(k,i),dat(k,j)) = temp(dat(k,i),dat(k,j)) + (1/(mu(k,1)-1)). end if. end loop. end loop. compute coinc = coinc + temp. end loop. compute q = reshape(coinc, (nrow(coinc)\*ncol(coinc)), 1). compute q = csum(q > 0). compute nc = rsum(coinc). compute n = csum(nc). compute coinct = coinc. compute dmat = diag(coinc). compute nzero = csum(dmat > 0). compute bootm = nprs. compute nx = (dmat/n)&\*\*bootm. compute nx=rnd(btn\*csum(nx)). compute numone = 0. compute expect = coinc. loop i = 1 to uniq. loop j = 1 to uniq. do if (i = j). compute expect(i,j)=nc(i,1)\*(nc(j,1)-1)/(n-1). else if (i <> j). compute expect(i,j)=nc(i,1)\*nc(j,1)/(n-1). end if. end loop. end loop. loop z = 1 to (btn + 1). do if (z > 1). compute btalp(z,1)=1. compute rchfirst=-1. loop u = 1 to nobj. compute muloop=(mu(u,1)\*(mu(u,1)-1))/2. loop ppp= 1 to muloop. compute rchoose=trunc(uniform(1,1)\*nprs)+1. do if (ppp = 2 and rchfirst=rchoose). compute rchoose=trunc(uniform(1,1)\*nprs)+1. end if. compute rchfirst=rchoose. compute btalp(z,1)=btalp(z,1)-(er(rchoose,1)/(mu(u,1)-1)). end loop. end loop. do if (btalp(z,1) <= -1). compute btalp(z,1)=-1. end if. end if. do if (z = 1). do if ( 2 = 2). compute delta = make(uniq,uniq,0). loop i = 1 to uniq. loop j = i to uniq. do if (i <> j). compute delta(i,j) = (csum(nc(i:j,1))-(nc(i,1)/2)-(nc(j,1)/2))\*\*2. compute delta(j,i) = delta(i,j). end if. end loop. end loop. compute v = {'Ordinal'}. do if (z = 1). compute deltat = delta. end if. end if. do if ( 2 = 1). compute delta = 1-ident(uniq). compute v = {'Nominal'}. compute deltat = delta. end if. do if ( 2 = 3). loop i = 1 to uniq. loop j = i to uniq. do if (i <> j). compute delta(i,j) = (map(i,1)-map(j,1))\*\*2. compute delta(j,i) = delta(i,j). end if. end loop. end loop. compute v = {'Interval'}. compute deltat = delta. end if. do if ( 2 = 4). loop i = 1 to uniq. loop j = i to uniq. do if (i <> j). compute delta(i,j) = ((map(i,1)-map(j,1))/(map(i,1)+map(j,1)))\*\*2. compute delta(j,i) = delta(i,j). end if. end loop. end loop. compute v = {'Ratio'}. compute deltat = delta. end if. compute num = csum(rsum(delta&\*coinc)). compute den = csum(rsum(delta&\*expect)). do if (den > 0). compute alp = 1-(num/den). compute btalp(1,1)=alp. compute expdis=csum(rsum((expect&\*delta)))/n. end if. compute er=make(nprs,3,0). compute cnt=0. loop k = 1 to nrow(dat). loop i = 1 to (ncol(dat)-1). loop j = (i+1) to ncol(dat). compute v1=dat(k,i). compute v2=dat(k,j). do if (v1 <> -9999999 and v2 <> -9999999). compute cnt=cnt+1. compute er(cnt,1:2)={v1,v2}. compute er(cnt,3)=delta(v1,v2). end if. end loop. end loop. end loop. compute er=er(:,3). loop i = 1 to nprs. compute er(i,1)=(2\*er(i,1))/(expdis\*csum(mu)). end loop. end if. end loop. compute alpfirst = btalp(1,1). do if (btn > 0). compute btalp=btalp(2:nrow(btalp),1). compute btalptmp = btalp. compute btalptmp(GRADE(btalp)) = btalp. compute btalp = btalptmp. compute btalp = btalp(1:nrow(btalp),1). compute mn = csum(btalp)/btn. compute low95 = trunc(.025\*btn). compute high95 = trunc(.975\*btn)+1. compute low95 = btalp(low95,1). compute high95 = btalp(high95,1). compute median = btalp(0.50\*btn). compute q = {.9, 0; .8, 0; .7, 0; 0.67, 0; .6, 0; .5, 0}. loop i = 1 to 6. compute qcomp = (btalp < q(i,1)). compute qcomp = csum(qcomp)/btn. compute q(i,2)=qcomp. end loop. end if. do if (btalp(1,1) = -999). compute btprob = 1. end if. print/title = 'Krippendorff''s Alpha Reliability Estimate'. do if (btn = 0 or btprob = 1). compute res = {alpfirst, nobj, nj, nprs}. compute lab = {'Alpha', 'Units', 'Obsrvrs', 'Pairs'}. end if. do if (btn > 0 and btprob = 0). compute res = {alpfirst, low95, high95, nobj, nj, nprs}. compute lab = {'Alpha', 'LL95%CI', 'UL95%CI', 'Units', 'Observrs', 'Pairs'}. end if. print res/title = ' '/rnames = v/cnames = lab/format = F10.4. do if (btn > 0 and btprob = 0). print q/title = 'Probability (q) of failure to achieve an alpha of at least alphamin:'/clabels = 'alphamin' 'q'/format = F10.4. print btn/title = 'Number of bootstrap samples:'. end if. print vn/title = 'Judges used in these computations:'/format = a8. do if ( 1 = 1). print/title = '===================================================='. print coinct/title = 'Observed Coincidence Matrix'/format = F9.2. print expect/title = 'Expected Coincidence Matrix'/format = F9.2. print deltat/title = 'Delta Matrix'/format F9.2. compute tmap = t(map). print tmap/title 'Rows and columns correspond to following unit values'/format = F9.2. end if. else. print/title = 'ERROR: Input Reliability Data Matrix Exhibits No Variation.'. end if. do if (btprob = 1). print/title = 'A problem was encountered when bootstrapping, so these results are not printed'. end if. print/title = 'Examine output for SPSS errors and do not interpret if any are found'. END MATRIX. |
| Resources | Processor Time | 00:01:18.29 |
| Elapsed Time | 00:01:20.00 |

Run MATRIX procedure:Krippendorff's Alpha Reliability Estimate Alpha LL95%CI UL95%CI Units Observrs PairsOrdinal .8029 .7588 .8444 1080.0000 2.0000 1080.0000Probability (q) of failure to achieve an alpha of at least alphamin: alphamin q .9000 1.0000 .8000 .4180 .7000 .0000 .6700 .0000 .6000 .0000 .5000 .0000Number of bootstrap samples: 10000Judges used in these computations: Coder1 Coder2====================================================Observed Coincidence Matrix 1582.00 76.00 76.00 426.00Expected Coincidence Matrix 1272.49 385.51 385.51 116.49Delta Matrix .00 1166400.0 1166400.0 .00Rows and columns correspond to following unit values .00 1.00Examine output for SPSS errors and do not interpret if any are found------ END MATRIX -----