

Constructing Bootstrap Confidence Intervals for Pearson's r

Zhongmin Cui, Dongmei Li, and Wei Tao

November 24, 2008

1 Introduction

Pearson's r is the statistic to evaluate the strength of the linear relationship between two normally distributed variables. However, since the theoretic sampling distribution of r is unknown, it is impossible to conduct hypothesis testing or establish a confidence interval (CI) directly using the observed r . One of the prevailing methods to test hypothesis or to establish a confidence interval is to use the Fisher's z transformation, the distribution of which is normal and has a known standard error equal to $1/\sqrt{N-3}$, where N is the number of paired cases.

The purposes of this project are to 1) construct CIs for r using the bootstrap method, 2) construct CIs for r using the traditional Fisher's z method, 3) compare the average estimate of r and the average width of the CIs between the bootstrap and Fisher's z methods, and 4) evaluate the coverage of the two methods. As a reference for comparison, Monte Carlo (MC) sampling distributions will be constructed to approximate the true sampling distribution of r .

2 Methodology

We adopt the following notations,

- θ : true parameter (true correlations),
- $\hat{\theta}$: observed correlation of a MC sample,
- $\hat{\theta}^*$: observed correlation of a bootstrap sample,
- r : Pearson's r correlation,
- S : total number of Monte Carlo samples, and
- B : total number of bootstrap samples.
- n : sample size - number of paired cases.

2.1 Data Simulation

To simulate paired data with correlation equal to θ , two factors are considered - the absolute value of the true correlation (θ) and the sample size. We will choose three levels of θ ($\theta = 0, 0.5, 0.95$) and four levels of sample size ($n = 20, 50, 100, 200$). In total, there are 12 simulation conditions (3×4).

For each simulation condition, three methods are used to estimate θ and the 95% CI - the MC method, the bootstrap method, and the Fisher's z method.

2.2 The Monte Carlo Confidence Interval

Since no theoretical sampling distribution is available, we will use the Monte Carlo (MC) procedure to approximate the true sampling distribution of Pearson's r . For a specific value of θ and n , 20000 ($S=20000$) random samples of size n will be generated from the bi-variate normal distribution with mean vector of $\mathbf{0}$ and variance-covariance matrix of

$$\begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix}.$$

For each random sample, calculate the observed correlation ($\hat{\theta}$).

The MC expected θ is the mean of the 20000 $\hat{\theta}$. The 95% MC CI is calculated as:

$$[CDF^{-1}(\alpha/2), CDF^{-1}(1 - \alpha/2)]$$

.

The width of the CI is:

$$CDF^{-1}(1 - \alpha/2) - CDF^{-1}(\alpha/2)$$

.

The coverage of the MC CI is exactly equal to 0.95.

2.3 The Bootstrap Confidence Interval

For each simulation condition, generate 1000 random samples (referred to as the real samples hereafter) from the same bi-variate normal distribution as the one used in the MC analysis. For each real sample, generate an unbiased bootstrap estimate of θ , letting $B=2000$, and construct a bootstrap CI using the percentile method. The average estimate of bootstrap $\hat{\theta}$ and average width among the 1000 real samples will be reported. The coverage of the bootstrap method is computed as the number of intervals containing θ divided by 1000.

2.4 The Fisher's z Confidence Interval

For each simulation condition, use the same 1000 real samples as those used in the Bootstrap analysis. Calculate the observed r for each real sample and transform r into Fisher's z using equation:

$$z_r = 0.5 \ln \frac{1+r}{1-r}.$$

The standard error of the Fisher's z transformation (denoted as SE_{fz}) is equal to $1/\sqrt{n-3}$, where n is the sample size. The Fisher's z CI is calculated using the following equation:

$$[z_r - \Phi(\alpha/2) * SE_{fz}, z_r + \Phi(1 - \alpha/2) * SE_{fz}].$$

Then, transform the lower and upper limit of z_r back to r using:

$$r = \frac{e^{2z_r} - 1}{e^{2z_r} + 1}$$

The width of the 95% CI is the difference between the upper limit and lower limit of r . The mean of the 1000 observed r s and widths will be computed. The coverage of the Fisher's z test is calculated as the number of intervals containing θ divided by 1000.

3 Results

3.1 MC Sampling Distribution of r

Table 1 presents the 95% confidence interval of r using the MC sampling distribution under each simulation condition. Figure 1 presents the same information in chart. As shown, the width of CI decreases as n increases for a given r . This finding is not surprising because a larger sample size is associated with a more accurate estimate of θ . Table 1 and Figure 1 also show that for a given sample size, the width decreases as theta increases (or, approaches extreme). This is an interesting finding because it suggests that the stronger the relationship it is, the more confident we are on what have we found. The results also show that when sample size is small ($n \leq 50$), the CI becomes less symmetric around the expected θ as r increases.

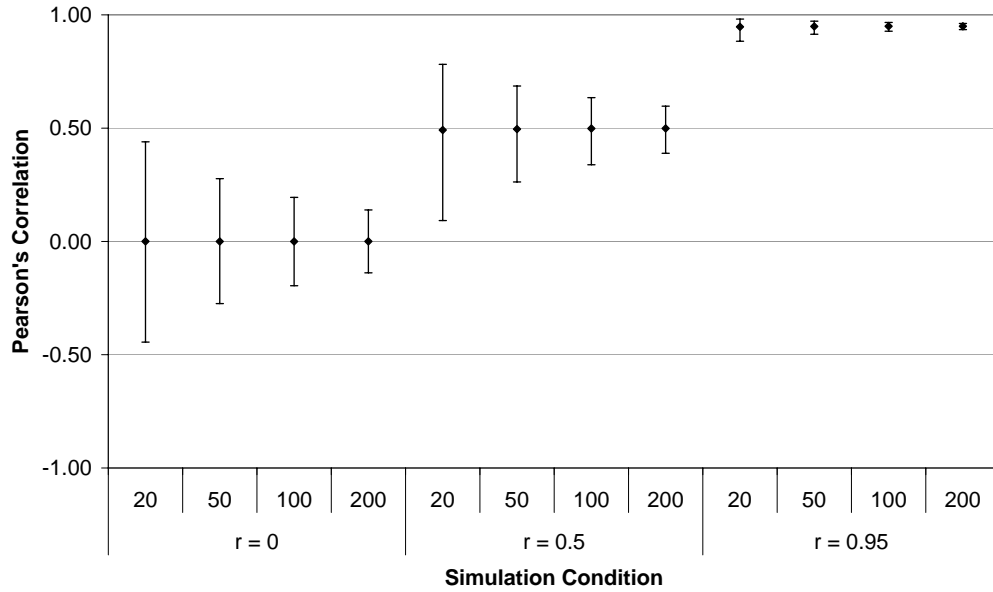
3.2 The bootstrap and Fisher's z CI

Table 2 presents the simulation result summary for all the three methods. For the bootstrap (BT) and Fisher's z (FZ) methods, the same pattern was found as that in the MC method. That is - the width of CI decreases as n increases for a given r , and the width of CI decreases as r increases for a given sample size. Also shown in Table 2 is that both the BT method and the FZ method yield

Table 1: 95% CI from the MC Sampling Distribution

θ	n	$\hat{\theta}$	Lower Limit	Upper Limit	Width
0	20	0.000	-0.444	0.439	0.884
	50	0.000	-0.274	0.277	0.551
	100	0.000	-0.196	0.194	0.390
	200	0.000	-0.139	0.139	0.278
0.5	20	0.492	0.092	0.782	0.689
	50	0.496	0.263	0.686	0.423
	100	0.498	0.338	0.634	0.296
	200	0.499	0.389	0.597	0.208
0.95	20	0.947	0.884	0.982	0.098
	50	0.949	0.915	0.972	0.058
	100	0.950	0.928	0.967	0.039
	200	0.950	0.935	0.962	0.027

Figure 1: 95% Confidence Interval around the Expected r , from the MC Sampling Distribution



reasonably accurate estimate of θ . The FZ method has a larger width than the BT method. As a result, the FZ method has a larger coverage than the BT method. The differences between the two methods on the width and on the coverage, however, become smaller as the sample size increases or as θ approaches 1. It appears that the coverage of the FZ method is not affected by the sample size and is closer to the nominal coverage than the BT method, especially when the sample size is small.

Table 2: Simulation Result Summary

θ	n	$\hat{\theta}$			Width			Coverage		
		MC	BT	FZ	MC	BT	FZ	MC	BT	FZ
0	20	0.000	-0.003	-0.003	0.884	0.839	0.846	0.950	0.926	0.950
	50	0.000	-0.003	-0.002	0.551	0.540	0.546	0.950	0.932	0.944
	100	0.000	-0.003	-0.003	0.390	0.386	0.389	0.950	0.946	0.953
	200	0.000	0.000	0.000	0.278	0.275	0.276	0.950	0.958	0.962
0.5	20	0.492	0.500	0.491	0.689	0.658	0.675	0.950	0.920	0.945
	50	0.496	0.502	0.499	0.423	0.411	0.419	0.950	0.923	0.936
	100	0.498	0.501	0.499	0.296	0.292	0.296	0.950	0.944	0.959
	200	0.499	0.501	0.500	0.208	0.206	0.208	0.950	0.943	0.948
0.95	20	0.947	0.951	0.948	0.098	0.099	0.107	0.950	0.921	0.940
	50	0.949	0.950	0.949	0.058	0.057	0.060	0.950	0.938	0.944
	100	0.950	0.950	0.950	0.039	0.039	0.040	0.950	0.938	0.944
	200	0.950	0.950	0.950	0.027	0.027	0.028	0.950	0.942	0.944

4 Conclusion

In summary, both the bootstrap and the Fisher's z methods perform well in this study. Based on the results, one might recommend the Fisher's z method in practice because the coverage of the Fisher's z CI is closer to the nominal coverage and is not affected by sample size. The computation is far less intensive than the bootstrap method. However, data in this study may favor the Fisher's z method because they were generated from a bivariate normal distribution. We suggest a future study with non-normal data.

Appendix: R codes

```
library(boot)
library(MASS)
library(QRMLib)

set.seed(1111)

#### function to compute the MC mean, se, and CI for Pearson's r

MCCI <- function (S, n, rho, alpha){

mc.r <- rep(NA, S)

  for (i in 1:S)
  {
    mvnorm.data <- rmnorm(n, Sigma=equicorr(2, rho), mu=rep(0, 2))
    mc.r[i] <- cor.test(mvnorm.data[,1], mvnorm.data[,2], method="pearson")$estimate
  }

mc.mean <- mean(mc.r)
mc.se <- sd(mc.r)
mc.l <- quantile(sort(mc.r), probs=alpha/2 )
mc.u <- quantile(sort(mc.r), probs=1-alpha/2 )

mc.width <- mc.u-mc.l

list(mc.mean=mc.mean, mc.l=mc.l, mc.u=mc.u, mc.width=mc.width)
}

#### function to compute Pearson r and CI using bootstrap

# function to compute Pearson correlation

corr <- function(dat, ind) {
  if (!(is.matrix( dat) &&
    ncol(dat) == 2 &&
    length(ind)== nrow(dat) ))
  {
    stop("invalid arguments")
  }
  cor.test(dat[ind, 1], dat[ind, 2], method="pearson")$estimate
}

boot.corr <- function (sample, r, replication = 2000, alpha = 0.05) {
```

```

boot.out <- boot(as.matrix(sample), corr, replication)
boot.ci.out <- boot.ci(boot.out, conf=1-alpha, type = "perc")
r_unbiased <- 2 * boot.out$t0 - mean(boot.out$t)
r_ci_width <- boot.ci.out$p[1,5] - boot.ci.out$p[1,4]
if(r < boot.ci.out$p[1,4] || r > boot.ci.out$p[1,5])
r_coverage <- 0
else r_coverage <- 1
list(r = r_unbiased, width = r_ci_width, coverage = r_coverage)
}

##### Function to compute Pearson r and CI using the Fisher'z method

fz <- function (data,alpha,r){

# data = the input data with two columns
# alpha = type I error rate
# r = true parameter

rhat <- cor.test(data[,1],data[,2],method="pearson")$estimate
zr <- 0.5 * log ( (1+rhat) / (1-rhat)) # The transformed Fisher's z
se <- 1/sqrt( nrow(data) -3) # Standard error of the Fisher's z distribution

zr.l <- zr - se * abs ( qnorm (alpha/2) ) # Lower limit of Fisher's z
zr.u <- zr + se * abs ( qnorm (alpha/2) ) # Upper limit of Fisher's z

rhat.l <- ( exp ( 2* zr.l) - 1 ) / ( exp (2*zr.l) + 1 ) # Convert zr.l to r
rhat.u <- ( exp ( 2* zr.u) - 1 ) / ( exp (2*zr.u) + 1 ) # Convert zr.u to r

width <- rhat.u - rhat.l # Width of the CI for r

if ( rhat.l < r & rhat.u > r) coverage <- 1 else coverage <- 0

list (rhat=rhat, width=width, coverage=coverage, rhat.l=rhat.l, rhat.u=rhat.u)
}

#####
#### Simulation function to generate MC results
#####

mcr.simu <- function (s,r,n){

# s = number of random real samples
# r = a vector of true r's
# n = a vector of sample sizes

```

```

mc.output <- numeric() # an empty variable to hold final mc results

for (j in 1:length(r)) # true r levels
{

  for (k in 1:length(n)) # sample size levels
  {
    mc.out <- MCCI (s, n[k], r[j], alpha=0.05)
    mc.output <- rbind ( mc.output,
      c( r[j], n[k], mc.out$mc.mean, mc.out$mc.l, mc.out$mc.u, mc.out$mc.width))
  }

}

colnames(mc.output)<-c("true.r","n","rhat","lower limit","upper limit","width")

list (mc.output= mc.output)
}

```

```

# Function to run the MC simulation
mcr.simu ( s=20000,r=c(0,0.5,0.95), n= c (20,50,100,200))

```

```

$mc.output

```

	true.r	n	rhat	lower limit	upper limit	width
[1,]	0.00	20	2.221782e-04	-0.44445892	0.4394914	0.88395030
[2,]	0.00	50	-1.861485e-04	-0.27443729	0.2769517	0.55138900
[3,]	0.00	100	-6.262504e-06	-0.19568526	0.1942863	0.38997153
[4,]	0.00	200	2.147865e-04	-0.13863497	0.1390569	0.27769186
[5,]	0.50	20	4.918014e-01	0.09217084	0.7815707	0.68939989
[6,]	0.50	50	4.957239e-01	0.26261793	0.6859687	0.42335075
[7,]	0.50	100	4.982670e-01	0.33828143	0.6344398	0.29615834
[8,]	0.50	200	4.989051e-01	0.38910942	0.5971746	0.20806516
[9,]	0.95	20	9.472231e-01	0.88357542	0.9815234	0.09794796
[10,]	0.95	50	9.491243e-01	0.91450495	0.9722258	0.05772085
[11,]	0.95	100	9.496028e-01	0.92764381	0.9665611	0.03891725
[12,]	0.95	200	9.497543e-01	0.93472271	0.9622173	0.02749460

```

#####
#### Simulation function to generate bootstrap and Fisher's z results
#####

```

```

btfz.simu <- function (s,r,n){

```

```

# s = number of random real samples
# r = a vector of true r's
# n = a vector of sample sizes

boot.output <- numeric() # an empty variable to hold final bootstrap output
fz.output <- numeric() # an empty variable to hold final fisher z output

for (j in 1:length(r)) # true r levels
{

  for (k in 1:length(n)) # sample size levels
  {
    boot.results <- numeric()
    fz.results <- numeric()

    for ( p in 1:s) # create s samples
    {
      data <- mvrnorm (n[k],c(0,0),matrix(c(1,r[j],r[j],1),ncol=2,byrow=T))

      boot.out <- boot.corr (data,r[j])

      boot.results <- rbind (boot.results,
        c ( boot.out$r, boot.out$width, boot.out$coverage ))

      fz.out <- fz ( data, alpha=0.05, r[j])
      fz.results <- rbind (fz.results,
        c ( fz.out$rhat, fz.out$width, fz.out$coverage))
    }

    boot.avg.rhat <- mean ( boot.results[,1])
    boot.avg.width <- mean ( boot.results [,2])
    boot.coverage <- sum ( boot.results [,3])/s

    fz.avg.rhat <- mean ( fz.results[,1])
    fz.avg.width <- mean ( fz.results [,2])
    fz.coverage <- sum ( fz.results [,3])/s

    boot.output <- rbind (boot.output,
      c ( r[j], n[k], boot.avg.rhat, boot.avg.width, boot.coverage) )

    fz.output <- rbind (fz.output,
      c ( r[j], n[k], fz.avg.rhat, fz.avg.width, fz.coverage) )

    colnames(boot.output) <- c ("true.r","n","avg.rhat","avg.width","coverage")
    colnames(fz.output) <- c ("true.r","n","avg.rhat","avg.width","coverage")
  }
}

```

```

    }
  }
list (boot.output = boot.output, fz.output = fz.output)
}

#Run the simulation
btfz.simu ( s=1000,r=c(0,0.5,0.95), n= c (20,50,100,200))

```

\$boot.output

	true.r	n	avg.rhat	avg.width	coverage
[1,]	0.00	20	-0.0027670659	0.83925513	0.926
[2,]	0.00	50	-0.0026519403	0.54029282	0.932
[3,]	0.00	100	-0.0027475795	0.38624749	0.946
[4,]	0.00	200	-0.0002695833	0.27495593	0.958
[5,]	0.50	20	0.5002035260	0.65776599	0.920
[6,]	0.50	50	0.5021771159	0.41132469	0.923
[7,]	0.50	100	0.5009767169	0.29202770	0.944
[8,]	0.50	200	0.5009886566	0.20593823	0.943
[9,]	0.95	20	0.9509082235	0.09874498	0.921
[10,]	0.95	50	0.9495493811	0.05732857	0.938
[11,]	0.95	100	0.9500659211	0.03934997	0.938
[12,]	0.95	200	0.9498710094	0.02736599	0.942

\$fz.output

	true.r	n	avg.rhat	avg.width	coverage
[1,]	0.00	20	-0.0026114865	0.84590383	0.950
[2,]	0.00	50	-0.0024956070	0.54595754	0.944
[3,]	0.00	100	-0.0027999078	0.38920533	0.953
[4,]	0.00	200	-0.0001544716	0.27616101	0.962
[5,]	0.50	20	0.4913946062	0.67457378	0.945
[6,]	0.50	50	0.4985033406	0.41939597	0.936
[7,]	0.50	100	0.4991534289	0.29568031	0.959
[8,]	0.50	200	0.5000736808	0.20828782	0.948
[9,]	0.95	20	0.9482067012	0.10724083	0.940
[10,]	0.95	50	0.9485845409	0.05977365	0.944
[11,]	0.95	100	0.9495914969	0.03994983	0.944
[12,]	0.95	200	0.9496334659	0.02771404	0.944