Appendix B (Pornprasertmanit & Little, in press)

R Script

The following code is written for the R Statistical program (R Development Core Team, 2011). The following code allows user to put data with two variables (`data`). The code will find the bootstrap confidence intervals of the correlation between two variables, the skewness (g_1) of each variable, the excessive kurtosis (g_2) of each variable, the difference between magnitude of skewness, and the difference between the magnitudes of excessive kurtosis. To use this script, first, the user copy the following code to the R interface.

```
centralMoment <- function(x, ord) {
	if(ord < 2) stop("Central moment can be calculated for order 2 or more in an integer.")
	wm <- mean(x)
	result <- sum((x - wm)^(ord))/length(x)
	return(result)
}

kStat <- function(x, ord) {
	n <- length(x)
	if(ord == 1) {
		return(mean(x))
	} else if (ord == 2) {
		return(centralMoment(x, 2) * n / (n - 1))
	} else if (ord == 3) {
		return(centralMoment(x, 3) * n^2 / ((n - 1) * (n - 2)))
	} else if (ord == 4) {
		num1 <- n^2
		num2 <- (n + 1) * centralMoment(x, 4)
		num3 <- 3 * (n - 1) * centralMoment(x, 2)^2
		denom <- (n - 1) * (n - 2) * (n - 3)
		return((num1 * (num2 - num3))/denom)
	} else {
		stop("Order can be 1, 2, 3, or 4 only.")
	}
}

skew <- function(object, population=FALSE) {
	if(population) {
		return(centralMoment(object, 3)/(centralMoment(object, 2)^(3/2)))
	} else {
		est <- kStat(object, 3)/(kStat(object, 2)^(3/2))
		se <- sqrt(6/length(object))
		z <- est/se
		p <- (1 - pnorm(abs(z)))*2
		return(c("skew (g1)"=est, se=se, z=z, p=p))
	}
}

kurtosis <- function(object, population=FALSE) {
	if(population) {
		return((centralMoment(object, 4)/(centralMoment(object, 2)^2)) - 3)
	} else {
		est <- kStat(object, 4)/(kStat(object, 2)^(2))
		se <- sqrt(24/length(object))
		z <- est/se
		p <- (1 - pnorm(abs(z)))*2
		return(c("Excess Kur (g2)"=est, se=se, z=z, p=p))
```

```
        }
}

contain <- function(vec, value) { vec <- sort(vec); (vec[1] <= value) & (value <= vec[2]) }

direcDepen <- function(data, R=2000, conf=0.95, bonferroni=FALSE, type=c("perc", "bca"),
toMatrix=FALSE) {
        if(is.matrix(data)) data <- as.data.frame(data)
        if(!is.data.frame(data)) stop("Please provide the data with two variables in a matrix or
data frame format")
        if(ncol(data) != 2) stop("The dataset does not have exactly two variables")
        if(bonferroni) conf <- 1 - ((1 - conf)/7)
        library(boot)
        fun <- function(dat, g) {
                dat <- dat[g, ]
                r <- cor(dat)[1, 2]
                sk1 <- skew(dat[,1])[1]
                sk2 <- skew(dat[,2])[1]
                exKur1 <- kurtosis(dat[,1])[1]
                exKur2 <- kurtosis(dat[,2])[1]
                diff1 <- abs(sk1) - abs(sk2)
                diff2 <- abs(exKur1) - abs(exKur2)
                result <- c(r, sk1, sk2, diff1, exKur1, exKur2, diff2)
                names(result) <- c("r", "skew1", "skew2", "diffSkew", "exKur1", "exKur2",
"diffExKur")
                return(result)
        }
        result <- boot(data, fun, R=R, stype="i")
        ci <- lapply(as.list(1:7), function(i, result, conf, type) boot.ci(result, conf=conf,
type=type, t0=result$t0[i], t=result$t[,i]), result=result, conf=conf, type=type)
        names(ci) <- c("r", "skew1", "skew2", "diffSkew", "exKur1", "exKur2", "diffExKur")
        if(length(type) == 1 && type == "all") type <- c("normal", "basic", "student", "percent",
"bca")
        type <- gsub("perc", "percent", type)
        type <- gsub("norm", "normal", type)
        type <- gsub("stud", "student", type)
        out <- lapply(as.list(type), function(object, type2) sapply(object, function(obj, type)
getElement(obj, type), type=type2), object=ci)
        out <- lapply(out, function(obj) t(obj[(nrow(obj)-1):nrow(obj),]))
        if(toMatrix) {
                out <- do.call(cbind, out)
                colnames(out) <- as.vector(t(outer(type, c(".lower", ".upper"), paste, sep="")))
        } else {
                names(out) <- type
                out <- lapply(out, function(obj) {colnames(obj) <- c("lower", "upper");
return(obj)})
        }
        return(out)
}


decisionSkew <- function(out) {
        sigR <- !contain(out["r",], 0)
        sigSkew1 <- !contain(out["skew1",], 0)
        sigSkew2 <- !contain(out["skew2",], 0)
        sigDiff <- !contain(out["diffSkew",], 0)
        if(!sigR) return("Neither")
        if(!sigSkew1 & !sigSkew2) return("Undetermined")
        if(sigSkew1 & sigSkew2) {
                rPos <- out["r", 1] > 0
                skew1Pos <- out["skew1", 1] > 0
                skew2Pos <- out["skew2", 1] > 0
                if(rPos & (skew1Pos != skew2Pos)) return("Neither")
                if(!rPos & (skew1Pos == skew2Pos)) return("Neither")
        }
        if(!sigDiff) return("Undetermined")
        diffPos <- out["diffSkew", 1] > 0
        ifelse(diffPos, return("var1"), return("var2"))
}

decisionExKur <- function(out) {
```

```
        sigR <- !contain(out["r",], 0)
        sigExKur1 <- !contain(out["exKur1",], 0)
        sigExKur2 <- !contain(out["exKur2",], 0)
        sigDiff <- !contain(out["diffExKur",], 0)
        if(!sigR) return("Neither")
        if(!sigExKur1 & !sigExKur2) return("Undetermined")
        if(sigExKur1 & sigExKur2) {
                exKur1Pos <- out["exKur1", 1] > 0
                exKur2Pos <- out["exKur2", 1] > 0
                if(exKur1Pos != exKur2Pos) return("Neither")
        }
        if(!sigDiff) return("Undetermined")
        diffPos <- out["diffExKur", 1] > 0
        ifelse(diffPos, return("var1"), return("var2"))
}
```

This code contains multiple functions. Next, the user interacts with the functions `direcDepen`, `decisionSkew`, and `decisionExKur` which will call the other functions. For example, we use the `attitude` dataset provided by the R program to illustrate how to use these functions. The first two variables (rating and complaints) will be used to determine directional dependency.

```
data <- attitude[,c(1, 2)]
out <- direcDepen(data)
```

If the user types `out` in the R interface, the user will see a list that contains percentile (`percent`) and bias-corrected and accelerated (`bca`) bootstrap confidence interval. You may pick either confidence interval in determining directional dependency. In this example, the bias-corrected and accelerated bootstrap confidence interval is chosen:

```
outbca <- out$bca
```

The user then can determine directional dependency based on skewness and excessive kurtosis by `decisionSkew`, and `decisionExKur` functions, respectively, and use the selected confidence-interval output as the argument:

```
decisionSkew(outbca)

decisionExKur(outbca)
```

Supplementary Materials for

Pornprasertmanit, S., & Little, T. D. (in press). Determining directional dependency in causal associations. *International Journal of Behavioral Development*.