# Supplementary Material for:
# "Simple Ways to Interpret Effects in Modeling Ordinal Categorical Data"

in *Statistica Neerlandica* (2018), by A. Agresti and C. Tarantola

The R–package `erer` of Sun (2016) has a function *ocME* that supplies marginal effects at the mean, using output from the *polr* function. This function works only for logit and probit link functions. We provide here an extension of it (called *ocMEM*) that handles also log-log and complementary log-log link functions. Furthermore we provide a function called *ocAME*, based on the *ocME* function for average marginal effects, that handles complementary log-log, logit, log-log and probit link functions.

One should first estimate the model using the function *polr* of the MASS library, and then pass the resulting object to *ocMEM*() or *ocAME*() to obtain the marginal effects estimates.

For the log-log link and comulative log-log function we also provide the density and distribution functions (*p1gumbel* and *d1gumbel* for the log-log; *p1Gumbel* and *d1Gumbel* for the comulative log-log ). These functions should be uploaded before using *ocME* and *ocAME*.

Table S1: R function to obtain the distribution function for the log-log link.

```
-----------------------------------------------------------------------------------------------------
p1gumbel<-function (q, loc = 0, scale = 1, lower.tail = TRUE)
 {
    q <- (q - loc)/scale
    p <- exp(-exp(-q))
    if (!lower.tail)
        1 - p
    else p
}
-----------------------------------------------------------------------------------------------------
```

Table S2: R function to obtain the density function for the log-log link.

```
d1gumbel<-function (x, loc = 0, scale = 1, log = FALSE)
 {
    x <- (x - loc)/scale
    d <- log(1/scale) - x - exp(-x)
    if (!log)
        exp(d)
    else d
}
```

Table S3: R function to obtain the distribution function for cumulative log-log link

```
p1Gumbel<-function (q, loc = 0, scale = 1, lower.tail = TRUE)
 {
    q <- (q - loc)/scale
    p <- exp(-exp(q))
    if (lower.tail)
        1 - p
    else p
}
```

Table S4: R function to obtain the density function for the cumulative log-log link.

```
d1Gumbel<-function (x, loc = 0, scale = 1, log = FALSE)
{
    x <- -(x - loc)/scale
    d <- log(1/scale) - x - exp(-x)
    if (!log)
        exp(d)
    else d
}
```

Table S5: R function to obtain marginal effects at the mean.

```
-----------------------------------------------------------------------------------------------------------
ocMEM<-function (w, rev.dum = TRUE, digits = 3) {

# 1. Check inputs

    if (!inherits(w, "polr")) {
        stop("Need an ordered choice model from 'polr()'.\n")
    }
    if (w$method != "probit" & w$method != "logistic" & w$method != "loglog" & w$method != "cloglog") {
        stop("Need a probit or logit model.\n")
    }

# 2. Get data out

    lev <- w$lev
    J <- length(lev)
    x.name <- attr(x = w$terms, which = "term.labels")
    x2 <- w$model[, x.name]
    ww <- paste("~ 1", paste("+", x.name, collapse = " "), collapse = " ")
    x <- model.matrix(as.formula(ww), data = x2)[, -1]
    x.bar <- as.matrix(colMeans(x))
    b.est <- as.matrix(coef(w))
    K <- nrow(b.est)
    xb <- t(x.bar) %*% b.est
    z <- c(-10^6, w$zeta, 10^6)
    pfun <- switch(w$method, probit = pnorm, logistic = plogis, loglog = p1gumbel, cloglog = p1Gumbel)
    dfun <- switch(w$method, probit = dnorm, logistic = dlogis, loglog = d1gumbel, cloglog = d1Gumbel)
    V2 <- vcov(w)
    V3 <- rbind(cbind(V2, 0, 0), 0, 0)
    ind <- c(1:K, nrow(V3) - 1, (K + 1):(K + J - 1), nrow(V3))
    V4 <- V3[ind, ]
    V5 <- V4[, ind]

# 3. Calculate marginal effects (ME) # 3.1 ME value

    f.xb<-matrix(0,nrow=1,ncol=2)
    f.xb[1,1]<- dfun(z[1] - xb) - dfun(z[2] - xb)
    f.xb[1,2]<- dfun(z[J] - xb) - dfun(z[J+1] - xb)
    me<- b.est %*% f.xb
```

```
    colnames(me) <- paste("effect", lev[c(1,J)], sep = ".")

# 3.2 ME standard error

    se <- matrix(0, nrow = K, ncol = 2)
    ind<-0
    for (j in c(1,J)) {
        u1 <- c(z[j] - xb)
        u2 <- c(z[j + 1] - xb)
        if (w$method == "probit") {
            s1 <- -u1
            s2 <- -u2
        }
        else if (w$method == "logistic"){
            s1 <- 1 - 2 * pfun(u1)
            s2 <- 1 - 2 * pfun(u2)
        }
        else if (w$method == "loglog"){
            s1 <- exp(-u1)-1
            s2 <- exp(-u2)-1
        }
        else if (w$method == "cloglog"){
            s1 <- (1-exp(u1))
            s2 <- (1-exp(u2))
        }


        else {
        stop("Specified link not available.")
            }
        d1 <- dfun(u1) * (diag(1, K, K) - s1 * (b.est %*% t(x.bar)))
        d2 <- -1 * dfun(u2) * (diag(1, K, K) - s2 * (b.est %*%
            t(x.bar)))
        q1 <- dfun(u1) * s1 * b.est
        q2 <- -1 * dfun(u2) * s2 * b.est

        if(j==1)
        {
        dr <- cbind( d2, 0, q2)
        }
        else
        {dr <- cbind( d1, q1, 0)
```

```
        }
        V <- V5[c(1:K, K + j, K + j + 1), c(1:K, K + j, K + j +
            1)]
        cova <- dr %*% V %*% t(dr)
        ind<-ind+1
        se[, ind] <- sqrt(diag(cova))
    }
    colnames(se) <- paste("SE", lev[c(1,J)], sep = ".")
    rownames(se) <- colnames(x)

# 4. Revise ME and standard error for dummy variable

    if (rev.dum) {
        for (k in 1:K) {ind<-0
            if (identical(sort(unique(x[, k])), c(0, 1))) {
                for (j in c(1,J)) {
                  x.d1 <- x.bar
                  x.d1[k, 1] <- 1
                  x.d0 <- x.bar
                  x.d0[k, 1] <- 0
                  ua1 <- z[j] - t(x.d1) %*% b.est
                  ub1 <- z[j + 1] - t(x.d1) %*% b.est
                  ua0 <- z[j] - t(x.d0) %*% b.est
                  ub0 <- z[j + 1] - t(x.d0) %*% b.est
                  ind<-ind+1
                  me[k, ind] <- pfun(ub1) - pfun(ua1) - (pfun(ub0) -
                    pfun(ua0))
                  d1 <- (dfun(ua1) - dfun(ub1)) %*% t(x.d1) -
                    (dfun(ua0) - dfun(ub0)) %*% t(x.d0)
                  q1 <- -dfun(ua1) + dfun(ua0)
                  q2 <- dfun(ub1) - dfun(ub0)
                  dr <- cbind(d1, q1, q2)
                  V <- V5[c(1:K, K + j, K + j + 1), c(1:K, K +
                    j, K + j + 1)]
                  se[k, ind] <- sqrt(c(dr %*% V %*% t(dr)))
                }
            }
        }
    }

#5. Output
```

```
    z.value <- me/se
    p.value <- 2 * (pnorm(abs(z.value),lower.tail = FALSE))
    out <- list()
            out[[1]] <- round(cbind(effect = me[, 1], std.error = se[,
            1], z.value = z.value[, 1], p.value = p.value[, 1]),
            digits)
            out[[2]] <- round(cbind(effect = me[, 2], std.error = se[,
            2], z.value = z.value[, 2], p.value = p.value[, 2]),
            digits)
    names(out) <- paste("ME", lev[c(1,J)], sep = ".")
    result <- out
    class(result) <- "ocMEM"
    return(result)
}
```

---

Table S6: R function to obtain avarage marginal effects.

```
-----------------------------------------------------------------------------------------------------------
ocAME<-function (w, rev.dum = TRUE, digits = 3) {

# 1. Check inputs

    if (!inherits(w, "polr")) {
        stop("Need an ordered choice model from 'polr()'.\n")
    }
    if (w$method != "probit" & w$method != "logistic" & w$method != "loglog" & w$method != "cloglog") {
        stop("Need a probit or logit model.\n")
    }

# 2. Get data out

    lev <- w$lev
    J <- length(lev)
    x.name <- attr(x = w$terms, which = "term.labels")
    x2 <- w$model[, x.name]
    ww <- paste("~ 1", paste("+", x.name, collapse = " "), collapse = " ")
    x <- model.matrix(as.formula(ww), data = x2)[, -1]
    b.est <- as.matrix(coef(w))
    K <- nrow(b.est)
    xb <- x %*% b.est
    z <- c(-10^6, w$zeta, 10^6)
    pfun <- switch(w$method, probit = pnorm, logistic = plogis, loglog = p1gumbel, cloglog = p1Gumbel)
    dfun <- switch(w$method, probit = dnorm, logistic = dlogis, loglog = d1gumbel, cloglog = d1Gumbel)
    V2 <- vcov(w)
    V3 <- rbind(cbind(V2, 0, 0), 0, 0)
    ind <- c(1:K, nrow(V3) - 1, (K + 1):(K + J - 1), nrow(V3))
    V4 <- V3[ind, ]
    V5 <- V4[, ind]

# 3. Calculate average marginal effects (AME)
# 3.1 AME value

    vec1<-rep(1, nrow(x))
    f.xb<-matrix(0,nrow(x),2)
    f.xb[,1]<-dfun(vec1%*% t(z[1]) -  xb) -dfun(vec1%*% t(z[2]) - xb)
    f.xb[,2]<-dfun(vec1%*% t(z[J]) -  xb) -dfun(vec1%*% t(z[J+1]) - xb)
```

```
    f.xb1<-apply(f.xb,2,mean)
    me <- b.est %*% matrix(data = f.xb1, nrow = 1)
    colnames(me) <- paste("effect", lev[c(1,J)], sep = ".")

# 3.2 AME standard error

   se <- matrix(0, nrow = K, ncol = 2)
  ind<-0
     for (j in c(1,J)) {
        temp<-matrix(0, nrow = K, ncol = length(b.est)+2)

    for (k in 1:length(xb)){
            u1 <- c(z[j] - xb[k])
            u2 <- c(z[j + 1] - xb[k])

        if (w$method == "probit") {
            s1 <- -u1
            s2 <- -u2
        }
        else if (w$method == "logistic"){
            s1 <- 1 - 2 * pfun(u1)
            s2 <- 1 - 2 * pfun(u2)
        }
        else if (w$method == "loglog"){
            s1 <- exp(-u1)-1
            s2 <- exp(-u2)-1
        }
        else if (w$method == "cloglog"){
            s1 <- (1-exp(u1))
            s2 <- (1-exp(u2))
        }
        else {
        stop("Specified link not available.")
            }
        d1 <- dfun(u1) * (diag(1, K, K) - s1 * (b.est %*% t(x[k,])))
        d2 <- -1 * dfun(u2) * (diag(1, K, K) - s2 * (b.est %*%
            t(x[k,])))
        q1 <- dfun(u1) * s1 * b.est
        q2 <- -1 * dfun(u2) * s2 * b.est
        if(j==1)
        {
```

```
        drtemp <- cbind( d2, 0, q2)
        }
        else
        {drtemp <- cbind( d1, q1, 0)
        }
        temp<-temp+drtemp
        }
        dr<-temp/length(xb)
        V <- V5[c(1:K, K + j, K + j + 1), c(1:K, K + j, K + j +
            1)]
        cova <- dr %*% V %*% t(dr)
        ind<-ind+1
        se[, ind] <- sqrt(diag(cova))
    }
    colnames(se) <- paste("SE", lev[c(1,J)], sep = ".")
    rownames(se) <- colnames(x)

# 4. Revise AME and standard error for dummy variable.

   if (rev.dum) {
       for (k in 1:K) {ind<-0
           if (identical(sort(unique(x[, k])), c(0, 1))) {
               for (j in c(1,J)) {
                 x.d1 <- x
                 x.d1[,k] <- 1
                 x.d0 <- x
                 x.d0[,k] <- 0
                 ua1 <-vec1*z[j] - x.d1 %*% b.est
                 ub1 <-vec1*z[j + 1] - x.d1 %*% b.est
                 ua0 <-vec1*z[j] - x.d0 %*% b.est
                 ub0 <-vec1*z[j + 1] - x.d0 %*% b.est
                 ind<-ind+1
                 me[k, ind] <- mean(pfun(ub1) - pfun(ua1) - (pfun(ub0) -
                   pfun(ua0)))
                    temp<-0
                for (g in 1: nrow(x)){
                    d1 <- (dfun(ua1[g]) - dfun(ub1[g])) %*% t(x.d1[g,]) -
                        (dfun(ua0[g]) - dfun(ub0[g])) %*% t(x.d0[g,])
                            q1 <- -dfun(ua1[g]) + dfun(ua0[g])
                            q2 <- dfun(ub1[g]) - dfun(ub0[g])
                            drtemp <- cbind(d1, q1, q2)
```

```
                        temp<-temp+drtemp
                        }
                        dr<-temp/nrow(x)
                        V <- V5[c(1:K, K + j, K + j + 1), c(1:K, K +
                            j, K + j + 1)]
                        se[k, ind] <- sqrt(c(dr %*% V %*% t(dr)))
                }
            }
        }
    }

# 5. Output

    z.value <- me/se
    p.value <- 2 * (pnorm(abs(z.value),lower.tail = FALSE))
    out <- list()
            out[[1]] <- round(cbind(effect = me[, 1], std.error = se[,
            1], z.value = z.value[, 1], p.value = p.value[, 1]),
            digits)
            out[[2]] <- round(cbind(effect = me[, 2], std.error = se[,
            2], z.value = z.value[, 2], p.value = p.value[, 2]),
            digits)
    names(out) <- paste("ME", lev[c(1,J)], sep = ".")
    result<-out
    class(result) <- "ocAME"
    return(result)
}
-------------------------------------------------------------------------------------------------------
```