04_FullData

J Andres Gannon, Erik Gartzke, Jon Lindsay, and Peter Schram, Center for Peace and Security Studies

2020-11-18

Contents

Load data

Load the newly created data with all relevant covariates and subset to the European sample

```
df_full <- readRDS(paste0(here::here(), '/data/grayzone_model.rds'))

# Limit sample to just European states
df <- df_full %>%
  dplyr::filter(continent == "Europe")
```

Fixed polr function

We use the polr function from the MASS package to compute an ordered probit. The base version of the function in the R package contains an error that does not take the log of differences in the reposed zetas which results in an optimization error where vmmin is infinite. A fixed version of the function was created and is loaded below. For this reason, the polr function is not loaded from the MASS package, but instead from the function below. All secondary functions from the MASS package do work

```
# file MASS/R/polr.R
# copyright (C) 1994-2008 W. N. Venables and B. D. Ripley
# Use of transformed intercepts contributed by David Firth
#
```

```r
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 or 3 of the License
#  (at your option).
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  A copy of the GNU General Public License is available at
#  http://www.r-project.org/Licenses/
#
polr <- function(formula, data, weights, start, ..., subset,
                 na.action, contrasts = NULL, Hess = FALSE,
                 model = TRUE,
                 method = c("logistic", "probit", "cloglog", "cauchit"))
{
    logit <- function(p) log(p/(1 - p))

    fmin <- function(beta) {
        theta <- beta[pc + 1L:q]
        gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)
        eta <- offset
        if (pc > 0)
            eta <- eta + drop(x %*% beta[1L:pc])
        pr <- pfun(gamm[y + 1] - eta) - pfun(gamm[y] - eta)
        if (all(pr > 0))
            -sum(wt * log(pr))
        else Inf
    }

    gmin <- function(beta)
    {
        jacobian <- function(theta) { ## dgamma by dtheta matrix
            k <- length(theta)
            etheta <- exp(theta)
            mat <- matrix(0 , k, k)
            mat[, 1] <- rep(1, k)
            for (i in 2:k) mat[i:k, i] <- etheta[i]
            mat
        }
        theta <- beta[pc + 1L:q]
        gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)
        eta <- offset
        if(pc > 0) eta <- eta + drop(x %*% beta[1L:pc])
        pr <- pfun(gamm[y+1] - eta) - pfun(gamm[y] - eta)
        p1 <- dfun(gamm[y+1] - eta)
        p2 <- dfun(gamm[y] - eta)
        g1 <- if(pc > 0) t(x) %*% (wt*(p1 - p2)/pr) else numeric(0)
        xx <- .polrY1*p1 - .polrY2*p2
        g2 <- - t(xx) %*% (wt/pr)
        g2 <- t(g2) %*% jacobian(theta)
```

```r
        if(all(pr > 0)) c(g1, g2) else rep(NA, pc+q)
    }

    m <- match.call(expand.dots = FALSE)
    method <- match.arg(method)
    pfun <- switch(method, logistic = plogis, probit = pnorm,
                   cloglog = pgumbel, cauchit = pcauchy)
    dfun <- switch(method, logistic = dlogis, probit = dnorm,
                   cloglog = dgumbel, cauchit = dcauchy)
    if(is.matrix(eval.parent(m$data)))
        m$data <- as.data.frame(data)
    m$start <- m$Hess <- m$method <- m$model <- m$... <- NULL
    m[[1L]] <- as.name("model.frame")
    m <- eval.parent(m)
    Terms <- attr(m, "terms")
    x <- model.matrix(Terms, m, contrasts)
    xint <- match("(Intercept)", colnames(x), nomatch=0L)
    n <- nrow(x)
    pc <- ncol(x)
    cons <- attr(x, "contrasts") # will get dropped by subsetting
    if(xint > 0) {
        x <- x[, -xint, drop=FALSE]
        pc <- pc - 1
    } else warning("an intercept is needed and assumed")
    wt <- model.weights(m)
    if(!length(wt)) wt <- rep(1, n)
    offset <- model.offset(m)
    if(length(offset) <= 1) offset <- rep(0, n)
    y <- model.response(m)
    if(!is.factor(y)) stop("response must be a factor")
    lev <- levels(y)
    if(length(lev) <= 2) stop("response must have 3 or more levels")
    y <- unclass(y)
    q <- length(lev) - 1
    Y <- matrix(0, n, q)
    .polrY1 <- col(Y) == y
    .polrY2 <- col(Y) == y - 1
    if(missing(start)) {
      # try something that should always work -tjb
      u <- as.integer(table(y))
      u <- (cumsum(u)/sum(u))[1:q]
      zetas <-
         switch(method,
                "logistic"= qlogis(u),
                "probit"=   qnorm(u),
                "cauchit"=  qcauchy(u),
                "cloglog"=  -log(-log(u)) )
      s0 <- c(rep(0,pc),zetas[1],log(diff(zetas)))

##         # try logistic/probit regression on 'middle' cut
##         q1 <- length(lev) %/% 2
##         y1 <- (y > q1)
##         X <- cbind(Intercept = rep(1, n), x)
```

```r
##         fit <-
##             switch(method,
##                    "logistic"= glm.fit(X, y1, wt, family = binomial(), offset = offset),
##                    "probit" = glm.fit(X, y1, wt, family = binomial("probit"), offset = offset),
##                    ## this is deliberate, a better starting point
##                    "cloglog" = glm.fit(X, y1, wt, family = binomial("probit"), offset = offset),
##                    "cauchit" = glm.fit(X, y1, wt, family = binomial("cauchit"), offset = offset))
##         if(!fit$converged)
##             stop("attempt to find suitable starting values failed")
##         coefs <- fit$coefficients
##         if(any(is.na(coefs))) {
##             warning("design appears to be rank-deficient, so dropping some coefs")
##             keep <- names(coefs)[!is.na(coefs)]
##             coefs <- coefs[keep]
##             x <- x[, keep[-1L], drop = FALSE]
##             pc <- ncol(x)
##           }
##         spacing <- logit((1L:q)/(q+1)) # just a guess
##         if(method != "logistic") spacing <- spacing/1.7
##         gammas <- -coefs[1L] + spacing - spacing[q1]
##         thetas <- c(gammas[1L], log(diff(gammas)))
##         s0 <- c(coefs[-1L], thetas)
    } else if(length(start) != pc + q)
    stop("'start' is not of the correct length")
    else {
        s0 <- if(pc > 0) c(start[seq_len(pc+1)], log(diff(start[-seq_len(pc)])))
        else c(start[1L], log(diff(start)))
      }
    res <- optim(s0, fmin, gmin, method="BFGS", hessian = Hess, ...)
    beta <- res$par[seq_len(pc)]
    theta <- res$par[pc + 1L:q]
    zeta <- cumsum(c(theta[1L],exp(theta[-1L])))
    deviance <- 2 * res$value
    niter <- c(f.evals=res$counts[1L], g.evals=res$counts[2L])
    names(zeta) <- paste(lev[-length(lev)], lev[-1L], sep="|")
    if(pc > 0) {
        names(beta) <- colnames(x)
        eta <- offset + drop(x %*% beta)
    } else eta <- offset + rep(0, n)

    cumpr <- matrix(pfun(matrix(zeta, n, q, byrow=TRUE) - eta), , q)
    fitted <- t(apply(cumpr, 1L, function(x) diff(c(0, x, 1))))
    dimnames(fitted) <- list(row.names(m), lev)
    fit <- list(coefficients = beta, zeta = zeta, deviance = deviance,
                fitted.values = fitted, lev = lev, terms = Terms,
                df.residual = sum(wt) - pc - q, edf = pc + q, n = sum(wt),
                nobs = sum(wt),
                call = match.call(), method = method,
        convergence = res$convergence, niter = niter, lp = eta)
    if(Hess) {
        dn <- c(names(beta), names(zeta))
        H <- res$hessian
        dimnames(H) <- list(dn, dn)
```

```r
        fit$Hessian <- H
    }
    if(model) fit$model <- m
    fit$na.action <- attr(m, "na.action")
    fit$contrasts <- cons
    fit$xlevels <- .getXlevels(Terms, m)
    class(fit) <- "polr"
    fit
}

print.polr <- function(x, ...)
{
    if(!is.null(cl <- x$call)) {
        cat("Call:\n")
        dput(cl, control=NULL)
    }
    if(length(coef(x))) {
        cat("\nCoefficients:\n")
        print(coef(x), ...)
    } else {
        cat("\nNo coefficients\n")
    }
    cat("\nIntercepts:\n")
    print(x$zeta, ...)
    cat("\nResidual Deviance:", format(x$deviance, nsmall=2), "\n")
    cat("AIC:", format(x$deviance + 2*x$edf, nsmall=2), "\n")
    if(nzchar(mess <- naprint(x$na.action))) cat("(", mess, ")\n", sep="")
    if(x$convergence > 0)
        cat("Warning: did not converge as iteration limit reached\n")
    invisible(x)
}

vcov.polr <- function(object, ...)
{
    jacobian <- function(theta) { ## dgamma by dtheta matrix
        k <- length(theta)
        etheta <- exp(theta)
        mat <- matrix(0 , k, k)
        mat[, 1] <- rep(1, k)
        for (i in 2:k) mat[i:k, i] <- etheta[i]
        mat
    }

    if(is.null(object$Hessian)) {
        message("\nRe-fitting to get Hessian\n")
    utils::flush.console()
        object <- update(object, Hess=TRUE,
                         start=c(object$coefficients, object$zeta))
    }
    vc <- MASS::ginv(object$Hessian)
    pc <- length(coef(object))
    gamma <- object$zeta
    z.ind <- pc + seq_along(gamma)
```

```r
    theta <- c(gamma[1L], log(diff(gamma)))
    J <- jacobian(theta)
    A <- diag(pc + length(gamma))
    A[z.ind, z.ind] <- J
    V <- A %*% vc %*% t(A)
    structure(V,  dimnames = dimnames(object$Hessian))
}

summary.polr <- function(object, digits = max(3, .Options$digits - 3),
                         correlation = FALSE, ...)
{
    cc <- c(coef(object), object$zeta)
    pc <- length(coef(object))
    q <- length(object$zeta)
    coef <- matrix(0, pc+q, 3, dimnames=list(names(cc),
                                c("Value", "Std. Error", "t value")))
    coef[, 1] <- cc
    vc <- vcov(object)
    coef[, 2] <- sd <- sqrt(diag(vc))
    coef[, 3] <- coef[, 1]/coef[, 2]
    object$coefficients <- coef
    object$pc <- pc
    object$digits <- digits
    if(correlation)
        object$correlation <- (vc/sd)/rep(sd, rep(pc+q, pc+q))
    class(object) <- "summary.polr"
    object
}

print.summary.polr <- function(x, digits = x$digits, ...)
{
    if(!is.null(cl <- x$call)) {
        cat("Call:\n")
        dput(cl, control=NULL)
    }
    coef <- format(round(x$coefficients, digits=digits))
    pc <- x$pc
    if(pc > 0) {
        cat("\nCoefficients:\n")
        print(x$coefficients[seq_len(pc), , drop=FALSE], quote = FALSE,
              digits = digits, ...)
    } else {
        cat("\nNo coefficients\n")
    }
    cat("\nIntercepts:\n")
    print(coef[(pc+1):nrow(coef), , drop=FALSE], quote = FALSE,
          digits = digits, ...)
    cat("\nResidual Deviance:", format(x$deviance, nsmall=2), "\n")
    cat("AIC:", format(x$deviance + 2*x$edf, nsmall=2), "\n")
    if(nzchar(mess <- naprint(x$na.action))) cat("(", mess, ")\n", sep="")
    if(!is.null(correl <- x$correlation)) {
        cat("\nCorrelation of Coefficients:\n")
        ll <- lower.tri(correl)
```

```r
        correl[ll] <- format(round(correl[ll], digits))
        correl[!ll] <- ""
        print(correl[-1, -ncol(correl)], quote = FALSE, ...)
    }
    invisible(x)
}

predict.polr <- function(object, newdata, type=c("class","probs"), ...)
{
    if(!inherits(object, "polr")) stop("not a \"polr\" object")
    type <- match.arg(type)
    if(missing(newdata)) Y <- object$fitted
    else {
        newdata <- as.data.frame(newdata)
        Terms <- delete.response(object$terms)
        m <- model.frame(Terms, newdata, na.action = function(x) x,
                         xlev = object$xlevels)
        if (!is.null(cl <- attr(Terms, "dataClasses")))
            .checkMFClasses(cl, m)
        X <- model.matrix(Terms, m, contrasts = object$contrasts)
        xint <- match("(Intercept)", colnames(X), nomatch=0L)
        if(xint > 0) X <- X[, -xint, drop=FALSE]
        n <- nrow(X)
        q <- length(object$zeta)
        eta <- drop(X %*% object$coefficients)
        pfun <- switch(object$method, logistic = plogis, probit = pnorm,
                       cloglog = pgumbel, cauchit = pcauchy)
        cumpr <- matrix(pfun(matrix(object$zeta, n, q, byrow=TRUE) - eta), , q)
        Y <- t(apply(cumpr, 1L, function(x) diff(c(0, x, 1))))
        dimnames(Y) <- list(rownames(X), object$lev)
    }
    if(missing(newdata) && !is.null(object$na.action))
        Y <- napredict(object$na.action, Y)
    switch(type, class = {
        Y <- factor(max.col(Y), levels=seq_along(object$lev),
                    labels=object$lev)
    }, probs = {})
    drop(Y)
}

extractAIC.polr <- function(fit, scale = 0, k = 2, ...)
{
    edf <- fit$edf
    c(edf, deviance(fit) + k * edf)
}

model.frame.polr <- function(formula, ...)
{
    dots <- list(...)
    nargs <- dots[match(c("data", "na.action", "subset"), names(dots), 0)]
    if(length(nargs) || is.null(formula$model)) {
        m <- formula$call
        m$start <- m$Hess <- m$... <- NULL
```

```r
        m[[1L]] <- as.name("model.frame")
        m[names(nargs)] <- nargs
        if (is.null(env <- environment(formula$terms))) env <- parent.frame()
        data <- eval(m, env)
        if(!is.null(mw <- m$weights)) {
            nm <- names(data)
            nm[match("(weights)", nm)] <- as.character(mw)
            names(data) <- nm
        }
        data
    } else formula$model
}

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

dgumbel <- function (x, loc = 0, scale = 1, log = FALSE)
{
    x <- (x - loc)/scale
    d <- log(1/scale) - x - exp(-x)
    d[is.nan(d)] <- -Inf                # -tjb
    if (!log) exp(d) else d
}

anova.polr <- function (object, ..., test = c("Chisq", "none"))
{
    test <- match.arg(test)
    dots <- list(...)
    if (length(dots) == 0L)
        stop('anova is not implemented for a single "polr" object')
    mlist <- list(object, ...)
    nt <- length(mlist)
    dflis <- sapply(mlist, function(x) x$df.residual)
    s <- order(dflis, decreasing = TRUE)
    mlist <- mlist[s]
    if (any(!sapply(mlist, inherits, "polr")))
        stop('not all objects are of class "polr"')
    ns <- sapply(mlist, function(x) length(x$fitted.values))
    if(any(ns != ns[1L]))
        stop("models were not all fitted to the same size of dataset")
    rsp <- unique(sapply(mlist, function(x) paste(formula(x)[2L])))
    mds <- sapply(mlist, function(x) paste(formula(x)[3L]))
    dfs <- dflis[s]
    lls <- sapply(mlist, function(x) deviance(x))
    tss <- c("", paste(1L:(nt - 1), 2:nt, sep = " vs "))
    df <- c(NA, -diff(dfs))
    x2 <- c(NA, -diff(lls))
    pr <- c(NA, 1 - pchisq(x2[-1L], df[-1L]))
    out <- data.frame(Model = mds, Resid.df = dfs, Deviance = lls,
```

```r
                      Test = tss, Df = df, LRtest = x2, Prob = pr)
    names(out) <- c("Model", "Resid. df", "Resid. Dev", "Test",
                    "   Df", "LR stat.", "Pr(Chi)")
    if (test == "none") out <- out[, 1L:6]
    class(out) <- c("Anova", "data.frame")
    attr(out, "heading") <-
        c("Likelihood ratio tests of ordinal regression models\n",
          paste("Response:", rsp))
    out
}

polr.fit <- function(x, y, wt, start, offset, method)
{
    logit <- function(p) log(p/(1 - p))

    fmin <- function(beta) {
        theta <- beta[pc + 1L:q]
        gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)
        eta <- offset
        if (pc > 0)
            eta <- eta + drop(x %*% beta[1L:pc])
        pr <- pfun(gamm[y + 1] - eta) - pfun(gamm[y] - eta)
        if (all(pr > 0))
            -sum(wt * log(pr))
        else Inf
    }

    gmin <- function(beta)
    {
        jacobian <- function(theta) { ## dgamma by dtheta matrix
            k <- length(theta)
            etheta <- exp(theta)
            mat <- matrix(0 , k, k)
            mat[, 1] <- rep(1, k)
            for (i in 2:k) mat[i:k, i] <- etheta[i]
            mat
        }
        theta <- beta[pc + 1L:q]
        gamm <- c(-Inf, cumsum(c(theta[1L], exp(theta[-1L]))), Inf)
        eta <- offset
        if(pc > 0) eta <- eta + drop(x %*% beta[1L:pc])
        pr <- pfun(gamm[y+1] - eta) - pfun(gamm[y] - eta)
        p1 <- dfun(gamm[y+1] - eta)
        p2 <- dfun(gamm[y] - eta)
        g1 <- if(pc > 0) t(x) %*% (wt*(p1 - p2)/pr) else numeric(0)
        xx <- .polrY1*p1 - .polrY2*p2
        g2 <- - t(xx) %*% (wt/pr)
        g2 <- t(g2) %*% jacobian(theta)
        if(all(pr > 0)) c(g1, g2) else rep(NA, pc+q)
    }

    pfun <- switch(method, logistic = plogis, probit = pnorm,
                   cloglog = pgumbel, cauchit = pcauchy)
```

```r
    dfun <- switch(method, logistic = dlogis, probit = dnorm,
                   cloglog = dgumbel, cauchit = dcauchy)
    n <- nrow(x)
    pc <- ncol(x)
    lev <- levels(y)
    if(length(lev) <= 2L) stop("response must have 3 or more levels")
    y <- unclass(y)
    q <- length(lev) - 1L
    Y <- matrix(0, n, q)
    .polrY1 <- col(Y) == y
    .polrY2 <- col(Y) == y - 1L
    # pc could be 0.
    s0 <- if(pc > 0) c(start[seq_len(pc+1)], diff(start[-seq_len(pc)]))
    else c(start[1L], diff(start))
    res <- optim(s0, fmin, gmin, method="BFGS")
    beta <- res$par[seq_len(pc)]
    theta <- res$par[pc + 1L:q]
    zeta <- cumsum(c(theta[1L],exp(theta[-1L])))
    deviance <- 2 * res$value
    names(zeta) <- paste(lev[-length(lev)], lev[-1L], sep="|")
    if(pc > 0) {
        names(beta) <- colnames(x)
        eta <- drop(x %*% beta)
    } else {
        eta <- rep(0, n)
    }
    list(coefficients = beta, zeta = zeta, deviance = deviance)
}

profile.polr <- function(fitted, which = 1L:p, alpha = 0.01,
                         maxsteps = 10, del = zmax/5, trace = FALSE, ...)
{
    Pnames <- names(B0 <- coefficients(fitted))
    pv0 <- t(as.matrix(B0))
    p <- length(Pnames)
    if(is.character(which)) which <- match(which, Pnames)
    summ <- summary(fitted)
    std.err <- summ$coefficients[, "Std. Error"]
    mf <- model.frame(fitted)
    n <- length(Y <- model.response(mf))
    O <- model.offset(mf)
    if(!length(O)) O <- rep(0, n)
    W <- model.weights(mf)
    if(length(W) == 0L) W <- rep(1, n)
    OriginalDeviance <- deviance(fitted)
    X <- model.matrix(fitted)[, -1L, drop=FALSE] # drop intercept
    zmax <- sqrt(qchisq(1 - alpha, 1))
    profName <- "z"
    prof <- vector("list", length=length(which))
    names(prof) <- Pnames[which]
    start <- c(fitted$coefficients, fitted$zeta)
    for(i in which) {
        zi <- 0
```

```
        pvi <- pv0
        Xi <- X[,  - i, drop = FALSE]
        pi <- Pnames[i]
        for(sgn in c(-1, 1)) {
            if(trace) {
                message("\nParameter:", pi, c("down", "up")[(sgn + 1)/2 + 1])
                utils::flush.console()
            }
            step <- 0
            z <- 0
            ## LP is the linear predictor including offset.
            ## LP <- X %*% fitted$coef + O
            while((step <- step + 1) < maxsteps && abs(z) < zmax) {
                bi <- B0[i] + sgn * step * del * std.err[i]
                o <- O + X[, i] * bi
                fm <- polr.fit(x = Xi, y = Y, wt = W, start = start[-i],
                               offset = o, method = fitted$method)
                ri <- pv0
                ri[, names(coef(fm))] <- coef(fm)
                ri[, pi] <- bi
                pvi <- rbind(pvi, ri)
                zz <- fm$deviance - OriginalDeviance
                if(zz > - 1e-3) zz <- max(zz, 0)
                else stop("profiling has found a better solution, so original fit had not converged")
                z <- sgn * sqrt(zz)
                zi <- c(zi, z)
            }
        }
        si <- order(zi)
        prof[[pi]] <- structure(data.frame(zi[si]), names = profName)
        prof[[pi]]$par.vals <- pvi[si, ]
    }
    val <- structure(prof, original.fit = fitted, summary = summ)
    class(val) <- c("profile.polr", "profile")
    val
}

confint.polr <- function(object, parm, level = 0.95, trace = FALSE, ...)
{
    pnames <- names(coef(object))
    if(missing(parm)) parm <- seq_along(pnames)
    else if(is.character(parm))  parm <- match(parm, pnames, nomatch = 0L)
    message("Waiting for profiling to be done...")
    utils::flush.console()
    object <- profile(object, which = parm, alpha = (1. - level)/4.,
                      trace = trace)
    confint(object, parm=parm, level=level, trace=trace, ...)
}

confint.profile.polr <-
  function(object, parm = seq_along(pnames), level = 0.95, ...)
{
    of <- attr(object, "original.fit")
```

```r
    pnames <- names(coef(of))
    if(is.character(parm))  parm <- match(parm, pnames, nomatch = 0L)
    a <- (1-level)/2
    a <- c(a, 1-a)
    pct <- paste(round(100*a, 1), "%")
    ci <- array(NA, dim = c(length(parm), 2L),
                dimnames = list(pnames[parm], pct))
    cutoff <- qnorm(a)
    for(pm in parm) {
        pro <- object[[ pnames[pm] ]]
        if(length(pnames) > 1L)
            sp <- spline(x = pro[, "par.vals"][, pm], y = pro[, 1])
        else sp <- spline(x = pro[, "par.vals"], y = pro[, 1])
        ci[pnames[pm], ] <- approx(sp$y, sp$x, xout = cutoff)$y
    }
    drop(ci)
}

logLik.polr <- function(object, ...)
    structure(-0.5 * object$deviance, df = object$edf, class = "logLik")

simulate.polr <- function(object, nsim = 1, seed = NULL, ...)
{
    if(!is.null(object$model) && any(model.weights(object$model) != 1))
        stop("weighted fits are not supported")

    rgumbel <- function(n, loc = 0, scale = 1) loc - scale*log(rexp(n))

    ## start the same way as simulate.lm
    if(!exists(".Random.seed", envir = .GlobalEnv, inherits = FALSE))
        runif(1)                     # initialize the RNG if necessary
    if(is.null(seed))
        RNGstate <- get(".Random.seed", envir = .GlobalEnv)
    else {
        R.seed <- get(".Random.seed", envir = .GlobalEnv)
    set.seed(seed)
        RNGstate <- structure(seed, kind = as.list(RNGkind()))
        on.exit(assign(".Random.seed", R.seed, envir = .GlobalEnv))
    }
    rfun <- switch(object$method, logistic = rlogis, probit = rnorm,
                   cloglog = rgumbel, cauchit = rcauchy)
    eta <- object$lp
    n <- length(eta)
    res <- cut(rfun(n*nsim, eta),
               c(-Inf, object$zeta, Inf),
               labels = colnames(fitted(object)),
               ordered_result = TRUE)
    val <- split(res, rep(seq_len(nsim), each=n))
    names(val) <- paste("sim", seq_len(nsim), sep="_")
    val <- as.data.frame(val)
    if (!is.null(nm <- rownames(fitted(object)))) row.names(val) <- nm
    attr(val, "seed") <- RNGstate
    val
```

```
}
```

European sample

Create year dummy columns for the year fixed effects

```
df <- fastDummies::dummy_cols(df, select_columns = 'year')
```

Model 1

For the baseline model we just look at the relationship between the intensity of Russia intervention and whether the target is a NATO member as well as logged minimum distance from Russia. These are our two independent variables of interest. We use year fixed effects and cluster standard errors at the country level.

```
# Select model 1 variables
df_m1 <- df %>%
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, dplyr::starts_with("year_"))

m1 <- polr(intensity ~ .,
          data = df_m1,
          method = "probit",
          Hess = TRUE)

# Country-clustered SE
m1 <- lmtest::coeftest(m1, vcov = sandwich::vcovCL(m1, factor(df$cname1)))

# Odds ratio
m1_or <- exp(coef(m1))
stargazer::stargazer(m1,
                     coef = list(m1_or),
                     omit = "year",
                     title = "Model 1 Odds Ratios",
                     p.auto = FALSE,
                     type = "text")
```

```
##
## Model 1 Odds Ratios
## ===========================================
##                    Dependent variable:
##                ---------------------------
##
## -------------------------------------------
## NATOmem_MEM1              0.769
##                          (0.218)
##
## lnmindistkm_rus          0.911***
##                          (0.035)
##
## ===========================================
## ===========================================
## Note:           *p<0.1; **p<0.05; ***p<0.01
```

Old model 1:

```
# Prep matrix inversion option for model
d <- rms::datadist(df)
options(datadist = "d")

# Model
m1 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 year,
               data = df,
               x = TRUE,
               y = TRUE)

## country-clustered standard errors
m1 <- rms::robcov(m1, df$cabbrev1)

# Show results
texreg::screenreg(m1)

ggstatsplot::ggcoefstats(m1,
                         title = "Model 1",
                         package = "ggsci",
                         only.significant = TRUE,
                         palette = "category20c_d3",
                         )

ggeffects::ggpredict(m1) %>%
  plot()
```

Model 2

Independent variables are NATO membership and logged minimum distance from Russia. Controls include logged GDP per capita, logged population, democracy, and nuclear status. We use year fixed effects.

```
# Select model 2 variables
df_m2 <- df %>%
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, demo1, nuclear1, lnpop1, lngdppc1_2010const, dp

# Model
m2 <- polr(intensity ~ .,
                 data = df_m2,
                 method = "probit",
                 Hess = TRUE)

# Country-clustered SE
m2 <- lmtest::coeftest(m2, vcov = sandwich::vcovCL(m2, factor(df$cname1)))

# Odds ratio
m2_or <- exp(coef(m2))
stargazer::stargazer(m2,
                     coef = list(m2_or),
                     title = "Model 2 Odds Ratios",
```

```r
                      omit = "year",
                      p.auto = FALSE,
                      type = "text")
```

```
##
## Model 2 Odds Ratios
## ==============================================
##                        Dependent variable:
##                    ---------------------------
##
## ----------------------------------------------
## NATOmem_MEM1                  0.869
##                              (0.197)
##
## lnmindistkm_rus             0.916***
##                              (0.028)
##
## demo11                        1.288
##                              (0.387)
##
## nuclear11                    2.905**
##                              (0.449)
##
## lnpop1                       1.215**
##                              (0.076)
##
## lngdppc1_2010const          0.648***
##                              (0.112)
##
## ==============================================
## ==============================================
## Note:              *p<0.1; **p<0.05; ***p<0.01
```

Old model 2:

```r
# Prep model
d <- rms::datadist(df)
options(datadist = "d")

# Model
m2 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 demo1 +
                 nuclear1 +
                 lnpop1 +
                 gdppc1_2010const +
                 year,
                data = df,
                x = TRUE,
                y = TRUE)

## country-clustered standard errors
m2 <- rms::robcov(m2, df$cabbrev1)
```

```r
# Show results
texreg::screenreg(m2)

ggstatsplot::ggcoefstats(m2,
                         title = "Model 2",
                         package = "ggsci",
                         only.significant = TRUE,
                         palette = "category20c_d3",
                         )

ggeffects::ggpredict(m2) %>%
  plot()
```

Model 3

For the final model, we include controls for CINC ratio, democracy, nuclear status, and civil war with year fixed effects and country-clustered standard errors.

```r
# Select model 2 variables
df_m3 <- df %>%
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, demo1, nuclear1, lnpop1, lngdppc1_2010const, c

# Model
m3 <- polr(intensity ~ .,
           data = df_m3,
           method = "probit",
           Hess = TRUE)

# Country-clustered SE
m3 <- lmtest::coeftest(m3, vcov = sandwich::vcovCL(m3, factor(df$cname1)))

# Odds ratio
m3_or <- exp(coef(m3))
stargazer::stargazer(m3,
                     coef = list(m3_or),
                     omit = "year",
                     title = "Model 3 Odds Ratios",
                     p.auto = FALSE,
                     type = "text")
```

```
##
## Model 3 Odds Ratios
## ==============================================
##                         Dependent variable:
##                     ---------------------------
##
## ----------------------------------------------
## NATOmem_MEM1                   1.344
##                               (0.269)
##
## lnmindistkm_rus              0.834***
##                               (0.049)
##
```

```
## demo11                         1.290
##                               (0.503)
##
## nuclear11                    15.289***
##                               (0.780)
##
## lnpop1                         1.066
##                               (0.174)
##
## lngdppc1_2010const           0.336***
##                               (0.184)
##
## cinc_ratio                     0.123
##                               (2.523)
##
## ===============================================
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Old model 3:

```r
# Exclude post-2012 observations since missing control variables (cinc) cause a singular information ma
df_m3 <- df

df_m3$year <- as.integer(df_m3$year) + 1993

df_m3 <- df_m3 %>%
  dplyr::filter(year < 2013) %>%
  dplyr::mutate(year = as.factor(year))

# Reset datadist so levels match
d <- rms::datadist(df_m3)
options(datadist = "d")

# Model
m3 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 demo1 +
                 nuclear1 +
                 cinc_ratio +
                 gdppc1_2010const +
                 year,
               data = df_m3,
               x = TRUE,
               y = TRUE)

## country-clustered standard errors
m3 <- rms::robcov(m3, df_m3$cabbrev1)

# Show results
texreg::screenreg(m3)

ggstatsplot::ggcoefstats(m3,
                         title = "Model 3",
```

```
                          package = "ggsci",
                          only.significant = TRUE,
                          palette = "category20c_d3",
                          )

ggeffects::ggpredict(m3) %>%
  plot()
```

Relevant states sample

We run the same models as above on a different sample. Here, we limit sample to European states that meet any 1 of the following criteria:

1. **History of conflict** – European states that have had a MID or ICB incident with Russia/the Soviet Union from 1945-1994.
2. **Former Soviet Union/Warsaw Pact** – European states that were formerly members of either the Soviet Union or Warsaw Pact.
3. **Contiguity** – European states that are contiguous with Russia.

Model 4

Replicate model 1 with the new sample.

```
# Use same variables as model 1 but subset to
df_m4 <- df %>%
  dplyr::filter(relevant_conserv == 1)

df_m4vars <- df_m4 %>%
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, dplyr::starts_with("year_"))

# Model
m4 <- polr(intensity ~ .,
           data = df_m4vars,
           method = "probit",
           Hess = TRUE)

# Country-clustered SE
m4 <- lmtest::coeftest(m4, vcov = sandwich::vcovCL(m4, factor(df_m4$cname1)))

# Odds ratio
m4_or <- exp(coef(m4))
stargazer::stargazer(m4,
                     coef = list(m1_or),
                     omit = "year",
                     title = "Model 4 Odds Ratios",
                     p.auto = FALSE,
                     type = "text")
```

```
##
## Model 4 Odds Ratios
## ==============================================
```

```
##                        Dependent variable:
##                    ---------------------------
##
## -------------------------------------------
## NATOmem_MEM1              0.769**
##                           (0.252)
##
## lnmindistkm_rus            0.911
##                           (0.034)
##
## ===========================================
## ===========================================
## Note:              *p<0.1; **p<0.05; ***p<0.01
```

Old model:

```r
# Prep matrix inversion option for model
d <- rms::datadist(df_conserv)
options(datadist = "d")

# Model
m4 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 year,
               data = df_conserv,
               x = TRUE,
               y = TRUE)

## country-clustered standard errors
m4 <- rms::robcov(m4, df_conserv$cabbrev1)

# Show results
texreg::screenreg(m4)

ggstatsplot::ggcoefstats(m4,
                         title = "Model 4",
                         package = "ggsci",
                         only.significant = TRUE,
                         palette = "category20c_d3",
                         )

ggeffects::ggpredict(m4) %>%
  plot()
```

Model 5

Replicate model 2

```r
# Use same variables as model 1 but subset to
df_m5 <- df %>%
  dplyr::filter(relevant_conserv == 1)

df_m5vars <- df_m5 %>%
```

```r
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, demo1, nuclear1, lnpop1, lngdppc1_2010const, dp

# Model
m5 <- polr(intensity ~ .,
                 data = df_m5vars,
                 method = "probit",
                 Hess = TRUE)

# Country-clustered SE
m5 <- lmtest::coeftest(m5, vcov = sandwich::vcovCL(m5, factor(df_m5$cname1)))

# Odds ratio
m5_or <- exp(coef(m5))
stargazer::stargazer(m5,
                     coef = list(m5_or),
                     omit = "year",
                     title = "Model 5 Odds Ratios",
                     p.auto = FALSE,
                     type = "text")
```

```
##
## Model 5 Odds Ratios
## ==============================================
##                         Dependent variable:
##                     ---------------------------
##
## ----------------------------------------------
## NATOmem_MEM1                   1.073
##                               (0.416)
##
## lnmindistkm_rus                0.968
##                               (0.029)
##
## demo11                         1.099
##                               (0.468)
##
## nuclear11                      2.009
##                               (0.492)
##
## lnpop1                        1.139**
##                               (0.057)
##
## lngdppc1_2010const            0.634**
##                               (0.192)
##
## ==============================================
## ==============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Old model:

```r
# Prep model
d <- rms::datadist(df_conserv)
options(datadist = "d")
```

```r
# Model
m5 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 demo1 +
                 nuclear1 +
                 lnpop1 +
                 gdppc1_2010const +
                 year,
               data = df_conserv,
               x = TRUE,
               y = TRUE)

## country-clustered standard errors
m5 <- rms::robcov(m5, df_conserv$cabbrev1)

# Show results
texreg::screenreg(m5)

ggstatsplot::ggcoefstats(m5,
                         title = "Model 2",
                         package = "ggsci",
                         only.significant = TRUE,
                         palette = "category20c_d3",
                         )

ggeffects::ggpredict(m5) %>%
  plot()
```

Model 6

Replicate model 3 on the new sample

```r
# Use same variables as model 1 but subset to
df_m6 <- df %>%
  dplyr::filter(relevant_conserv == 1)

df_m6vars <- df_m6 %>%
  dplyr::select(intensity, NATOmem_MEM, lnmindistkm_rus, demo1, nuclear1, lnpop1, lngdppc1_2010const, c

# Model
m6 <- polr(intensity ~ .,
           data = df_m6vars,
           method = "probit",
           Hess = TRUE)

# Country-clustered SE
m6 <- lmtest::coeftest(m6, vcov = sandwich::vcovCL(m6, factor(df_m6$cname1)))

# Odds ratio
m6_or <- exp(coef(m6))
stargazer::stargazer(m6,
                     coef = list(m6_or),
```

```
                     omit = "year",
                     title = "Model 6 Odds Ratios",
                     p.auto = FALSE,
                     type = "text")
```

```
##
## Model 6 Odds Ratios
## ==============================================
##                       Dependent variable:
##                   ---------------------------
##
## ----------------------------------------------
## NATOmem_MEM1                  1.522
##                              (0.403)
##
## lnmindistkm_rus              0.909**
##                              (0.046)
##
## demo11                        0.992
##                              (0.586)
##
## nuclear11                    3.861*
##                              (0.813)
##
## lnpop1                        0.724
##                              (0.224)
##
## lngdppc1_2010const          0.363***
##                              (0.188)
##
## cinc_ratio                   36.396
##                              (3.761)
##
## ==============================================
## ==============================================
## Note:              *p<0.1; **p<0.05; ***p<0.01
```

Old model:

```
# Exclude post-2012 observations since missing control variables (cinc) cause a singular information ma
df_m6 <- df_conserv

df_m6$year <- as.integer(df_m6$year) + 1993

df_m6 <- df_m6 %>%
  dplyr::filter(year < 2013) %>%
  dplyr::mutate(year = as.factor(year))

# Model
m6 <- rms::lrm(intensity ~
                 NATOmem_MEM +
                 lnmindistkm_rus +
                 demo1 +
                 nuclear1 +
                 cinc_ratio +
```

```r
                 year,
               data = df_m6,
               x = TRUE,
               y = TRUE)

## country-clustered standard errors
m6 <- rms::robcov(m6, df_m6$cabbrev1)

# Show results
texreg::screenreg(m6)

ggstatsplot::ggcoefstats(m6,
                         title = "Model 6",
                         package = "ggsci",
                         only.significant = TRUE,
                         palette = "category20c_d3",
                         )

ggeffects::ggpredict(m6) %>%
  plot()
```

Compiled results

Compiled results of the 6 models are shown below:

```r
# Make list of models
models <- list(m1, m2, m3, m4, m5, m6)

# HTML version for markdown
texreg::screenreg(models,
                  stars = c(0.01, 0.05, 0.1, 0.15),
                  symbol = "\\dagger",
                  omit.coef = "(y)",
                  custom.coef.names = c("NATO member",
                                        "Russia min. distance",
                                        "Democracy",
                                        "Nuclear power",
                                        "Population",
                                        "GDP per capita",
                                        "CINC ratio"),
                  custom.gof.rows = list("Fixed effects" = c("Yes", "Yes", "Yes", "Yes", "Yes", "Yes")),
                  custom.header = list("Full sample" = 1:3, "Relevant states sample" = 4:6))
```

```
##
## ===========================================================================================
##                                     Full sample                  Relevant states sample
##                         -----------------------------------  ----------------------------------
##                         Model 1     Model 2     Model 3     Model 4     Model 5     Model 6
## -------------------------------------------------------------------------------------------
## NATO member              -0.26       -0.14        0.30       -0.53 **     0.07        0.42
##                          (0.22)      (0.20)      (0.27)      (0.25)      (0.42)      (0.40)
## Russia min. distance     -0.09 ***   -0.09 ***   -0.18 ***   -0.01       -0.03       -0.10 **
##                          (0.04)      (0.03)      (0.05)      (0.03)      (0.03)      (0.05)
```

```
## Democracy                             0.25        0.25                    0.09       -0.01
##                                      (0.39)      (0.50)                  (0.47)      (0.59)
## Nuclear power                         1.07 **     2.73 ***                0.70        1.35 *
##                                      (0.45)      (0.78)                  (0.49)      (0.81)
## Population                            0.19 **     0.06                    0.13 **    -0.32
##                                      (0.08)      (0.17)                  (0.06)      (0.22)
## GDP per capita                       -0.43 ***   -1.09 ***               -0.46 **    -1.01 ***
##                                      (0.11)      (0.18)                  (0.19)      (0.19)
## CINC ratio                                       -2.10                                3.59
##                                                  (2.52)                              (3.76)
## ---------------------------------------------------------------------------------------------
## Fixed effects          Yes          Yes          Yes          Yes         Yes         Yes
## =============================================================================================
## *** p < 0.01; ** p < 0.05; * p < 0.1; \dagger p < 0.15
```

```r
# Tex version for manuscript
texreg::texreg(models,
                stars = c(0.01, 0.05, 0.1, 0.15),
                symbol = "\\dagger",
                omit.coef = "(y)",
              custom.coef.names = c("NATO member",
                                    "Russia min. distance",
                                    "Democracy",
                                    "Nuclear power",
                                    "Population",
                                    "GDP per capita",
                                    "CINC ratio"),
                custom.gof.rows = list("Fixed effects" = c("Yes", "Yes", "Yes", "Yes", "Yes", "Yes")),
                custom.header = list("Full sample" = 1:3, "Relevant states sample" = 4:6),
               label = "table:model",
               caption = "All models are ordinal probits with year fixed effects and standard errors clu
               float.pos = "h",
               file = paste0(here::here(), "/paper/figures/","model.tex")
               )
```

```
## The table was written to the file '/home/andresgannon/Dropbox (rex)/Grad School/andres_github_private
```