**SUPPLEMENTARY MATERIAL**

**FULL R CODE FOR IMPLEMENTING QBA OF MISSING ECOG DATA AND OBTAINING E-VALUES FOR UNKNOWN CONFOUNDERS**

library(mice)

library(dplyr)

library(tableone)

library(WeightIt)

library(cobalt)

library(survey)

library(survminer)

# Please create a folder called "imputed\_data" to store imputed datasets before running

# Read data

df <- readRDS("data.rds")

#' Custom function to impute ordinal variables with delta adjustment

#' This function applies a variable `delta` which should be specified in

#' the global namespace when imputing ordinal variables with mice::mice()

mice.impute.deltaecog <- function(y, ry, x, wy = NULL, nnet.maxit = 100,

                                  nnet.trace = FALSE, nnet.MaxNWts = 1500,

                                  polr.to.loggedEvents = FALSE, ...) {

    if (is.null(wy)) wy <- !ry

    x <- as.matrix(x)

    aug <- mice:::augment(y, ry, x, wy)

    x <- aug$x

    y <- aug$y

    ry <- aug$ry

    wy <- aug$wy

    w <- aug$w

    xy <- cbind.data.frame(y = y, x = x)

    fit <- MASS::polr(formula(xy),

                      data = xy[ry, , drop = FALSE],

                      weights = w[ry],

                      control = list(...))

    # MNAR shift - add delta which is in the global namespace

    fit$zeta <- fit$zeta + delta

    post <- predict(fit, xy[wy, , drop = FALSE], type = "probs")

    if (sum(wy) == 1) {

        post <- matrix(post, nrow = 1, ncol = length(post))

    }

    fy <- as.factor(y)

    nc <- length(levels(fy))

    un <- rep(runif(sum(wy)), each = nc)

    if (is.vector(post)) {

        post <- matrix(c(1 - post, post), ncol = 2)

    }

    draws <- un > apply(post, 1, cumsum)

    idx <- 1 + apply(draws, 2, sum)

    levels(fy)[idx]

}

#######################

###

### MULTIPLE IMPUTATION

###

#######################

# Initialize mice

init <- mice(df, maxit=0)

# Can change predictor matrix if needed

pred <- init$pred

# Set delta adjustment method for ECOG

meth <- init$meth

meth["ecog"] <- "deltaecog"

#' Wrapper for mice() function that suppresses "non-integer" warnings

impute <- function(df, n=1, verbose=TRUE) {

    withCallingHandlers({

        #imp <- mice(df, meth=meth, pred=pred, m=n, print=verbose)

        imp <- mice(df, meth=meth, pred=pred, m=n, print=verbose)

    }, warning=function(w) {

        if (grepl("non-integer", w)) {

            invokeRestart("muffleWarning")

        } # non-integer success warnings are fine,

        # mice uses weighted regression so target is not integer

    })

    return(imp)

}

# Run imputations for a range of delta values and save these imputed files

print("Running imputations...")

deltas <- -5:5

for (delta in deltas) {

    message(paste0("On delta: ", delta))

    f <- paste0("imputed\_data/delta\_", delta, ".rds")

    imp <- impute(df, n=3)

    print(imp$loggedEvents)

    saveRDS(imp, f)

}

#######################

###

### EFFECT ESTIMATION

###

#######################

# Choose some parameters to estimate HRs for

treated <- "Pralsetinib"

refs <- c("Carboplatin,Pembrolizumab,Pemetrexed", "Pembrolizumab")

ref <- refs[1] # Either 1 or 2

# Formatted text for graphs etc

ref.form <- ifelse(ref=="Carboplatin,Pembrolizumab,Pemetrexed", "Pembro+chemo", "Pembrolizumab")

# Choose delta value to estimate HRs for

delta <- 0

# Helper function to create cohorts

cohortize <- function(x) {

    x %>%

        rename(drug=linename, OS=os, EVENT=event\_os) %>%

        filter(drug %in% c(treated, ref)) %>%

        filter(ecog %in% c(0, 1)) %>%

        droplevels()

}

# Specify confounders to adjust for

covs <- c("age65", "sex", "smokhist2", "tsd2", "stage2", "ecog", "race2")

# Helper function to run IPTW Cox model

iptw <- function(df, covs, ref, treated) {

    print(paste0("Reference: ", ref))

    print(paste0("Treated: ", treated))

    # Create treatment indicator

    df <- df %>%

        filter(drug %in% c(ref, treated)) %>%

        mutate(drug=ifelse(drug==treated, 1, 0)) %>%

        tidyr::drop\_na(all\_of(covs))

    # Compute tableone

    tabs <- CreateTableOne(vars=covs, strata="drug", data=df)

    print(tabs, smd=TRUE)

    # Compute propensity weights

    W <- weightit(f.build("drug", covs), data=df, method="ps", estimand="ATT", stabilize=FALSE)

    df$wts <- W$ps # get\_w\_from\_ps(W$ps, treat=df$drug, estimand="ATT")

    if (any(df$wts > 3)) {

        df$wts <- trim(df$wts, at=0.99)

    }

    print(summary(df$wts))

    # Compute re-weighted data

    svy <- svydesign(ids=~1, data=df, weights=~df$wts)

    svyt <- svyCreateTableOne(data = svy, strata = "drug", vars=covs)

    smds <- ExtractSmd(svyt)[,1]

    print(svyt, smd=T)

    if (any(smds>0.12)) {

        message("Imbalance: ", paste(names(smds[smds>0.1]), collapse=","))

        message("Imbalance: ", paste((smds[smds>0.1]) %>% round(3), collapse=","))

    }

    # Create KM curve

    km.fit <- do.call(survfit, list(Surv(OS, EVENT) ~ drug, data = df, weight = df$wts))

    gg <- suppressWarnings(survminer::ggsurvplot(km.fit,

                                                 palette=RColorBrewer::brewer.pal(3, "Set1"),

                                                 pval=TRUE,

                                                 pval.size=4.5,

                                                 size=0.7,

                                                 censor.size=4,

                                                 legend.labs=c(ref.form, treated),

                                                 xlab="Time (months)",

                                                 risk.table=T,

                                                 surv.median.line="hv",

                                                 pval.coord = c(45, 0.03),

                                                 conf.int=T))

    # Compute Cox model

    message("Cox regression")

    cox.fit <- coxph(as.formula(paste0("Surv(OS, EVENT) ~ drug + ", paste0(covs, collapse="+"))),

                     data=df, weights=df$wts, robust=TRUE)

    print(summary(coxph(as.formula(paste0("Surv(OS, EVENT) ~ drug + ", paste0(covs, collapse="+"))),

                        data=df, weights=NULL, robust=TRUE)))

    print(cox.fit)

    list(

        km=gg,

        surv=suppressWarnings(survminer::surv\_median(km.fit)),

        n0=nrow(df[df$drug==0,]),

        n1=nrow(df[df$drug==1,]),

        HR=cox.fit$coefficients[["drug"]],

        lower=confint(cox.fit)["drug",][[1]],

        upper=confint(cox.fit)["drug",][[2]],

        seHR=summary(cox.fit)$coefficients[1,4],

        coef=summary(cox.fit),

        meds=surv\_median(km.fit))

}

# Read data

tra <- readRDS("imputed\_data/delta\_0.rds") # Does not change with delta

eca <- readRDS(paste0("imputed\_data/delta\_", delta, ".rds")) # ECOG PS imputed by delta only in ECA

# For each imputation, concatenate trial arm and ECA and run IPTW-CoxPH

imp.results <- lapply(1:tra$m, function(j) {

    x0 <- cohortize(mice::complete(tra, j)) %>% filter(drug == ref)

    x1 <- cohortize(mice::complete(eca, j)) %>% filter(drug== treated)

    x <- rbind(x0, x1)

    iptw(x, covs, ref=ref, treated=treated)

})

# Check results

imp.results

# Function to pool estimates after imputation

# Other pooling approaches may exist

pool <- function(iptw.out, niters) {

    # Pool point estimates

    ests <- sapply(iptw.out, function(i) i$HR)

    est <- mean(ests)

    # Pooled within variance

    se <- sapply(iptw.out, function(i) i$seHR)

    var.wit <- mean(se\*\*2)

    # Pooled between variance

    var.bet <- 1/(niters-1) \* sum((est - ests)\*\*2)

    # Total variance

    var.tot <- var.wit + var.bet + var.bet/niters

    # Degrees of freedom

    lambda <- (var.bet + (var.bet/niters)) / var.tot

    df.old <- (niters-1)/(lambda^2)

    n <- iptw.out[[1]]$n1 + iptw.out[[1]]$n0

    df.obs <- ((n-1)+1)/((n-1)+3) \* (n-1) \* (1-lambda)

    df.adj <- (df.old \* df.obs) / (df.old + df.obs)

    # 95% confidence intervals

    cint <- qt(c(.025, .975), df=df.adj) \* (sqrt(var.tot)) + est

    # p-values

    tStat <- est / sqrt(var.tot)

    pval <- 2 \* pt(-abs(tStat), df=df.adj)

    # Mean median survival time

    medsurv <- sapply(iptw.out, function(i) i$meds$median) %>% rowMeans()

    lowsurv <- sapply(iptw.out, function(i) i$meds$lower) %>% rowMeans()

    uppsurv <- sapply(iptw.out, function(i) i$meds$upper) %>% rowMeans()

    # Return estimates

    list(HR=est, lower=cint[1], upper=cint[2], p=pval,

         median\_surv=medsurv, lower\_surv=lowsurv, upper\_surv=uppsurv)

}

# Results from imputation

out <- pool(imp.results, length(imp.results))

print(paste0("HR: ", round(exp(out$HR), 3), "; 95% CI: ", round(exp(out$lower), 3), "-", round(exp(out$upper), 3)))

#######################

###

### E-VALUES

###

#######################

# Function for computing E-values

evalue <- function(pt\_est,

                   lower=NA,

                   upper=NA,

                   type=c("HR", "OR", "RR"),

                   common=TRUE) {

    # Estimates

    ests <- c(pt\_est, lower, upper)

    # E-value computation

    .compute\_evalue <- function(x) {

        x <- ifelse(x < 1, 1/x, x)

        x + sqrt(x \* (x - 1))

    }

    type=match.arg(type)

    # Convert effect measures to approximate risk ratios

    # if needed

    if (type=="RR") {

        rr <- ests

    } else if (type=="OR") {

        rr <- EValue::toRR(EValue::OR(ests, rare=!common))

    } else if (type=="HR") {

        rr <- EValue::toRR(EValue::HR(ests, rare=!common))

    }

    # Compute E-value from risk ratios

    e <- .compute\_evalue(rr)

    # Return results

    data.frame(

        at=c("point", "lower", "upper"),

        value=ests, # Original estimates

        RR=as.vector(rr), # Estimates as risk ratios

        e\_value=e # E-values

    )

}

# E-values

evalue(out$HR %>% exp(), out$lower %>% exp(), out$upper %>% exp())