APPENDIX 1 – Implementation of percentile bootstrap confidence intervals

The SAS, Stata, and R code shown below can be used to obtain percentile bootstrap confidence intervals for the standardized estimates calculated in Figure 1. The code below uses 2 000 samples, but this can be changed by the user.

```
SAS CODE
* Set up bootstrap resampling;
        data boot;
                do sample=1 to 2000;
                        do i=1 to nobs;
                        pt=round(ranuni(12)*nobs);
                        set one nobs=nobs point=pt ;
                        output;
                        end;
                end;
        stop; run;

        proc logistic data = boot descending; by sample;
                model E = Z1 Z2;
                output out = boot predicted=ps;

        proc logistic data = boot descending; by sample;
                model E = ;
                output out=boot predicted=marg_pr;

        data boot;
                set boot;
                sw = E*marg_pr/ps + (1-E)*(1-marg_pr)/(1-ps); run;

* Obtain bootstrap confidence intervals for risk ratio ;

        ods output Estimates=rr_est ;
        proc genmod data = boot descending; by sample;
                model D = E / link=log dist=bin ;
                weight sw; estimate 'rr' E 1 / exp; run;
                ods rtf close;

        data rr;
        set rr_est;
                if Label ne 'rr'; epred=LBetaEstimate; run;

        proc univariate data=rr;
                var epred; output out=rr_cis pctlpts=2.5 97.5 pctlpre=rr_cis; run;

        proc print data=rr_cis noobs label; run;

* Obtain bootstrap confidence intervals for risk difference ;

        ods output Estimates=rd_est ;
        proc genmod data = boot descending; by sample;
                model D = E / link=identity dist=bin ;
                weight sw; estimate 'rd' E 1;  run;
                ods rtf close;

        data rd;
        set rd_est;
                epred=LBetaEstimate; run;

        proc univariate data=rd;
                var epred; output out=rd_cis pctlpts=2.5 97.5 pctlpre=rd_cis; run;

        proc print data=rd_cis noobs label; run;


STATA CODE
*For the RR
        program margrr, rclass

*Calculate denominators used in inverse probability weights
        logit E Z1 Z2
        predict den

*Create stabilized weights, using a null model with E as the dependent variable
        logit E
        predict num
        g sw=E*num/den+(1-E)*(1-num)/(1-den)

*Fit a log binomial model to the weighted data for the E-D association, with robust
variance
        glm D E [pw=sw],family(binomial) link(log) robust
        matrix b=e(b)
        local b=el(b,1,1)
        return scalar beta =`b'
        drop num den sw
        end


        bootstrap b=r(beta): margrr
        estat bootstrap, eform

*For the RD
        program margrd, rclass

*Calculate denominators used in inverse probability weights
        logit E Z1 Z2
        predict den

*Create stabilized weights, using a null model with E as the dependent variable
        logit E
        predict num
        g sw=E*num/den+(1-E)*(1-num)/(1-den)

*Fit a linear binomial model to the weighted data for the E-D association, with robust
variance
        glm D E [pw=sw],family(binomial) link(id) robust
        matrix b=e(b)
        local b=el(b,1,1)
        return scalar beta =`b'
        drop num den sw
        end

        bootstrap b=r(beta): margrd
        estat bootstrap,


R CODE

    # If 'boot' package not installed, enter in console: install.packages('boot')
        library(boot)

    # Specify starting value for random number generation for re-sampling
        set.seed(12)

    # For the risk ratio, create wrapper function for bootstrap procedure...
    # in which the propensity score is re-estimated in each re-sampling
        sptw.wrap=function(dat,indices)
        {
        dat=dat[indices,]
        E.out=glm(E~Z1+Z2,family=binomial(link="logit"),data=dat,na.action=na.exc
lude)
        ps=predict(E.out,type="response")
        new.sptw=dat$E*mean(dat$E)/ps+(1-dat$E)*(1-mean(dat$E))/(1-ps)
        coef(glm(D~E,family=binomial(link="log"),weight=new.sptw,data=dat))[2]
        }

    # Invoke wrapper function to perform bootstrap using the dataset of interest for
2000 samples
        boot.out=boot(one,sptw.wrap,2000)
        boot.out

    # Display percentile bootstrap point and 95% confidence interval estimates
        median(boot.out$t)
        boot.ci(boot.out,type="perc",conf=0.95)

    # plot density of bootstrap resamples
        plot(density(boot.out$t))


    # For the risk difference, create wrapper function for bootstrap procedure...
    # in which the propensity score is re-estimated in each re-sampling
        sptw.wrap=function(dat,indices)
        {
        dat=dat[indices,]

        E.out=glm(E~Z1+Z2,family=binomial(link="logit"),data=dat,na.action=na.exc
lude)
        ps=predict(E.out,type="response")
        new.sptw=dat$E*mean(dat$E)/ps+(1-dat$E)*(1-mean(dat$E))/(1-ps)

        coef(glm(D~E,family=binomial(link="identity"),weight=new.sptw,data=dat))[2
]
        }

    # Invoke wrapper function to perform bootstrap using the dataset of interest for
2000 samples
        boot.out=boot(one,sptw.wrap,2000)
        boot.out

    # Display percentile bootstrap point and 95% confidence interval estimates
        median(boot.out$t)
        boot.ci(boot.out,type="perc",conf=0.95)

    # plot density of bootstrap resamples
        plot(density(boot.out$t))
```

APPENDIX 2 – Implementation of standardized (weighted) estimates for two dichotomous exposure variables of interest

SAS CODE

```
/*Calculate denominators for weights. Logistic regression model for SMK. */
        proc logistic data = EVANS descending;
        model smk = age age*age age*age*age;
        output out = outpssmk predicted=ps; run;

/*Fit a second logistic regression model with CAT as the dependent variable. */
        proc logistic data = EVANS descending;
        model cat = smk age age*age age*age*age chl chl*chl chl*chl*chl age*smk
age*age*smk age*age*age*smk;
        output out = outpscat predicted=pc; run;

/*Create one dataset with conditional probabilities for SMK (ps) and CAT (pc) for
each obs. */
        proc sort data = outpssmk; by id;  proc sort data = outpscat; by id; run;

        data margstruc;
        merge outpssmk outpscat; by id; run;

/*Create stabilized weights for SMK, first using a null model with SMK as dependent
variable.*/
        proc logistic data = margstruc descending;
        model smk = ; output out=iptw2 predicted =marg_pr_smk; run;

        data iptw2; set iptw2;
        swsmk = smk*marg_pr_smk/ps + (1-smk)*(1-marg_pr_smk)/(1-ps);run;

/*Create stabilized weights for CAT first using a null model with CAT as the
dependent variable*/
        proc logistic data = iptw2 descending;
        model cat =; output out=iptw2 predicted =marg_pr_cat; run;

        data iptw2; set iptw2;
        swcat = cat*marg_pr_cat/pc + (1-cat)*(1-marg_pr_cat)/(1-pc); run;

/*Compute final regression weights. */
        data iptw2; set iptw2;
        swfinal = swsmk*swcat; run;

/*Fit log binomial model for standardized risk ratios with robust variance. */
        proc genmod data = iptw2 descending;
        class id;
        model chd = cat smk cat*smk / link=log dist=bin covb;
        weight swfinal;
        repeated subject=id / type=ind; run;

/*Fit linear binomial model for standardized risk differences with robust variance. */
        proc genmod data = iptw2 descending;
        class id;
        model chd = cat smk cat*smk / link=identity dist=bin covb;
        weight swfinal;
        repeated subject=id / type=ind;
        run;
```

STATA CODE

```
*Calculate denominators for weights. Model for SMK
        logit smk  age age*age age*age*age
        predict ps

*Second model for CAT
        logit cat  smk age age*age age*age*age chl chl*chl chl*chl*chl age*smk
age*age*smk age*age*age*smk
        predict pc


*Create stabilized weights, using a null model with SMK as the dependent variable
        logit smk
        predict marg_prsmk
        g swsmk=smk*marg_prsmk/ps+(1-smk)*(1-marg_prsmk)/(1-ps)

*Create stabilized weights, using a null model with CAT as the dependent variable
        logit cat
        predict marg_prcat
        g swcat=cat*marg_prcat/pc+(1-cat)*(1-marg_prcat)/(1-pc)

 *Compute final regression weights
      g swfinal = swsmk*swcat

*Fit a log binomial model for standardized risk ratios with robust variance
        glm chd cat smk cat*smk [pw=swfinal],family(binomial) link(log) robust

*Fit a linear binomial model for standardized risk differences with robust variance
        glm chd cat smk cat*smk [pw=swfinal],family(binomial) link(id) robust
```

R CODE

```
# variable definitions:
    # age2 = age*age
    # age3 = age*age*age
    # chl2 = chl*chl
    # chl3 = chl*chl*chl
    # a1s = age*smk
    # a2s = age*age*smk
    # a3s = age*age*age*smk
    # c1s = cat*smk

# Calculate denominators for weights. Logistic regression model for SMK
    smk.out=glm(smk~age+age2+age3,family=binomial(link="logit"),data=evans,n
    a.action=na.exclude)
    ps=predict(smk.out,type="response")
    summary(smk.out)

# Fit a second logistic regression model with CAT as the dependent variable
    cat.out=glm(cat~smk+age+age2+age3+chl+chl2+chl3+a1s+a2s+a3s,family=bin
    omial(link="logit"),data=evans,na.action=na.exclude)
    pc=predict(cat.out,type="response")
    summary(cat.out)

# Create stabilized weights for SMK, using the "mean" operator to create
numerators
    swsmk=evans$smk*mean(evans$smk)/ps+(1-(evans$smk))*(1-
    mean(evans$smk))/(1-ps)
    summary(swsmk)

# Create stabilized weights for CAT, using the "mean" operator to create
numerators
    swcat=evans$cat*mean(evans$cat)/pc+(1-(evans$cat))*(1-
    mean(evans$cat))/(1-pc)
    summary(swcat)

# Compute final regression weights
    swfinal = swsmk*swcat
    summary(swfinal)

# Fit log binomial model for standardized risk ratios with robust variance
    library(geepack)
    summary(geeglm(chd~cat+smk+c1s,family=binomial(link="log"),
    weight=swfinal, id=id, data=evans))

# Fit linear binomial model for standardized risk differences with robust variance
    library(geepack)
    summary(geeglm(chd~cat+smk+c1s,family=binomial(link="identity"),
    weight=swfinal, id=id, data=evans))
```