Web Material

Multiple-Imputation Variance Estimation in Studies With Missing or Misclassified Inclusion Criteria

Mark J. Giganti and Bryan E. Shepherd

Contents

Web Appendix 1

Rubin's rules variance estimator

Suppose we have a dataset of n subjects; some of the observations have missing values. Multiple imputation is applied to account for the missing data. For the $k = 1, \ldots, m$ imputed datasets, we estimate the parameter of interest ($\hat{\beta}_k$) and the corresponding model variance estimate ($\hat{\sigma}_k^2$). The averaged parameter estimator across imputations is $\bar{\beta}_I = \frac{1}{m} \sum\limits_k \hat{\beta}_k$. The usual imputation variance estimator proposed by Rubin is defined as:

$$\hat{\sigma}_I^2 = \frac{1}{m} \sum_k \hat{\sigma}_k^2 + \left(1 + \frac{1}{m}\right) \frac{1}{m-1} \sum_k (\hat{\beta}_k - \bar{\beta}_I)^2$$

Robins and Wang variance estimator

Suppose we have a dataset of n subjects with missing values. The p parameter imputation model is fit by estimating $\hat{\theta}$ among the subjects with non-missing values and we generate m imputation datasets. For those $k = 1, \ldots, m$ imputed datasets, we estimate the parameter of interest ($\hat{\beta}_k$). The Robins and Wang imputation variance estimator Γ is defined as:

$$\Gamma = \frac{1}{n} \tau^{-1} \Delta (\tau^{-1})^T, \qquad \text{where}$$

$$\begin{aligned}
\Delta &= \Omega + \kappa \Lambda \kappa^T + \frac{1}{n} \sum_{i=1}^{n} \{\kappa d_i^T \bar{u}_i + (\kappa d_i^T \bar{u}_i)^T\}, \\
\Omega &= \frac{1}{n} \sum_{i=1}^{n} \bar{u}_i^T \bar{u}_i, \\
\Lambda &= \frac{1}{n} \sum_{i=1}^{n} d_i^T d_i, \\
\kappa &= \frac{1}{nm} \sum_{i=1}^{n} \sum_{k=1}^{m} (u_i(\hat{\theta}, \hat{\beta}_k))^T S_{i,k}^{mis}, \qquad \text{and} \\
\bar{u}_i^T &= \frac{1}{m} \sum_{k=1}^{m} u_i(\hat{\theta}, \hat{\beta}_k)^T
\end{aligned}$$

There are four terms (S^{mis}, d, u, and τ) in the formula above that need to be calculated based on either the imputation model or the analysis model. We briefly describe these components and their calculation below.

Imputation model components

Two of these components (S^{mis}, d) are derived from the imputation model, $f(x_i \,|\, \theta)$, where X generally includes multiple variables, some of which are missing for some i. Here $f(x_i \,|\, \theta)$ represents a joint model for all X, which may be specified as a set of conditional models. Values of (S^{mis}, d) are based on the score function and its derivative for the imputation model. Let $v_i = 1$ denote that the i^{th} observation was observed and $v_i = 0$ denote that the i^{th} observation was imputed. For the k^{th} imputation, S_k^{mis} is a $n \times p$ matrix corresponding to the score of each parameter in the imputation model evaluated for each observation that was imputed. However, if the observation was not imputed, a value of 0 is assigned.

$$\begin{aligned} S_{i,k}^{mis} &= \left[\frac{\partial log f(x_i \,|\, \theta)}{\partial \theta} \,|_{\,\theta=\hat{\theta}} \right] (1 - v_i) \\ S_{i,k}^{obs} &= \left[\frac{\partial log f(x_i \,|\, \theta)}{\partial \theta} \,|_{\,\theta=\hat{\theta}} \right] v_i \end{aligned}$$

To calculate d, we first take derivates of the score function with respect to each parameter, evaluate at each observation that was not imputed, and take the average. We then take the inverse of this $p \times p$ matrix, multiply by the transpose of S^{obs}, and multiply by -1. Note that S^{obs} is the score evaluated for each observation that was not imputed; imputed values are assigned a value of 0.

$$d_i^T = - \left[\frac{1}{n} \sum_{i=1}^{n} v_i \frac{\partial}{\partial \theta^T} \left(\frac{\partial log f(x_i \,|\, \theta)}{\partial \theta} \right) \,|_{\,\theta=\hat{\theta}} \right]^{-1} S_i^{obs\,T}$$

Analysis model components

The final two components (u, τ) are derived from the analysis model. Both values are based on the estimating equations pertaining to the analysis model. For each imputation, we evaluate the estimating equation $u_i(\hat{\theta}, \hat{\beta}_k)$ for all subjects in the analysis dataset. To calculate τ, we take the derivative of the estimating equation and evaluate as:

$$\tau = -\frac{1}{nm} \sum_{i=1}^{n} \sum_{k=1}^{m} \left(\frac{\partial u_{i,k}}{\partial \beta^T} \right) \,|_{\,\beta=\hat{\beta}}$$

Additional Intuition

The multiple components behind the Robins and Wang imputation variance estimator can be intimidating. To gain intuition, it might be helpful to first consider a simple scenario where we estimate the variance for an analysis where all values are observed and thus there is no imputation. When no values are imputed, $\Delta = \Omega$ since:

$$S^{mis} = 0 \implies \kappa = 0 \implies \Delta = \Omega$$

Suppose we had a linear regression analysis model for the association between two variables, Y and X. Thus, $u_i = X_i(Y_i - X_i\beta)$ and $\tau = \frac{1}{n}\sum_{i=1}^{n} X_i^T X_i$. The variance estimator is:

$$\begin{aligned}
\Gamma =& \frac{1}{n}\tau^{-1}\Omega(\tau^{-1})^T \\
=& \frac{1}{n}\left(\frac{1}{n}\sum X_i^T X\right)^{-1}\frac{1}{n}\sum X_i(Y_i - X_i\beta)X_i(Y - X_i\beta)\left(\frac{1}{n}\sum X_i^T X_i\right)^{-1} \\
=& \left(\sum X_i^T X_i\right)^{-1}\sum X_i(Y_i - X_i\beta)(Y_i - X_i\beta)X_i\left(\sum X_i^T X_i\right)^{-1}
\end{aligned}$$

We recognize this as the robust variance estimator for linear regression. In fact, the Robins and Wang variance estimator calculates the robust variance estimator when there is no imputed data.

When data are imputed, $\Delta \neq \Omega$ since $S^{mis} \neq 0$. As a result, the variance estimator calculation is no longer as simple as calculating the robust variance estimator based on the analysis model. This should make sense given the uncertainty from imputing values in addition to the uncertainty of generating estimates based on a sample. To better understand, it may be helpful to slightly re-arrange the Robins and Wang variance estimator as follows:

$$
\begin{aligned}
\Gamma =& \frac{1}{n}\tau^{-1}\Delta(\tau^{-1})^T \\
=& \frac{1}{n}\tau^{-1}\Omega(\tau^{-1})^T \qquad + \\
& \frac{1}{n}\tau^{-1}\kappa\Lambda\kappa^T(\tau^{-1})^T \qquad + \\
& \frac{1}{n}\tau^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\{\kappa d_i^T\bar{u}_i + (\kappa d_i^T\bar{u}_i)^T\}\right)(\tau^{-1})^T
\end{aligned}
$$

It should be emphasized that the second and third terms are calculated using components from both the imputation and analysis models. This contrasts with the calculation for the variance estimator proposed by Rubin that is based solely on the analysis model.

Web Table 1

Estimated coverage probabilities when imputation variance estimated using Rubin's approach for different validation subsample sizes (n= 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000) as well as different inclusion thresholds ($A > \{-\infty$, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3$\}$). These numbers correspond to Figure 1A.

Inclusion threshold	Analysis dataset size	Validation subsample size							
	(mean)	500	1,000	1,500	2,000	2,500	3,000	3,500	4,000
$A > 3$	524(13%)	1	1	1	0.998	0.996	0.986	0.975	0.955
$A >$ 2.5	791(20%)	0.995	0.998	0.998	0.995	0.991	0.983	0.971	0.948
$A > 2$	1,127(28%)	0.990	0.997	0.996	0.992	0.989	0.981	0.970	0.954
$A >$ 1.5	1,520(38%)	0.978	0.988	0.989	0.988	0.986	0.981	0.971	0.957
$A > 1$	1,947(49%)	0.965	0.978	0.981	0.982	0.980	0.977	0.968	0.958
$A >$ 0.5	2,379(59%)	0.960	0.970	0.974	0.974	0.974	0.972	0.962	0.949
$A > 0$	2,783(70%)	0.950	0.965	0.971	0.965	0.967	0.963	0.959	0.951
$A >$ -0.5	3,136(78%)	0.947	0.956	0.965	0.962	0.963	0.961	0.955	0.949
$A >$ -1	3,420(86%)	0.943	0.950	0.961	0.956	0.962	0.959	0.954	0.945
$A >$ -∞	4,000(100%)	0.942	0.948	0.955	0.952	0.957	0.956	0.955	0.944

Web Table 2

Estimated coverage probabilities when imputation variance estimated using Robins and Wang's approach for different validation subsample sizes (n= 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000) as well as different inclusion thresholds ($A > \{-\infty$, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3$\}$). These numbers correspond to Figure 1B.

Inclusion threshold	Analysis dataset size	Validation subsample size							
	(mean)	500	1,000	1,500	2,000	2,500	3,000	3,500	4,000
$A > 3$	524(13%)	0.932	0.936	0.939	0.942	0.948	0.948	0.948	0.952
$A > 2.5$	791(20%)	0.935	0.943	0.940	0.943	0.946	0.947	0.946	0.946
$A > 2$	1,127(28%)	0.938	0.943	0.943	0.945	0.952	0.952	0.951	0.953
$A > 1.5$	1,520(38%)	0.943	0.939	0.947	0.946	0.950	0.953	0.954	0.956
$A > 1$	1,947(49%)	0.943	0.940	0.948	0.948	0.952	0.955	0.952	0.955
$A > 0.5$	2,379(59%)	0.942	0.943	0.950	0.947	0.949	0.954	0.952	0.946
$A > 0$	2,783(70%)	0.943	0.944	0.948	0.944	0.948	0.953	0.950	0.946
$A >$ -0.5	3,136(78%)	0.942	0.942	0.949	0.945	0.948	0.952	0.946	0.943
$A >$ -1	3,420(86%)	0.942	0.942	0.947	0.940	0.948	0.949	0.946	0.940
$A >$ -∞	4,000(100%)	0.942	0.943	0.948	0.943	0.948	0.947	0.946	0.939

Web Appendix 2

Simulation example: R code for data generation

The following R code generates a dataset containing 4,000 electronic health records with four key variables: two continuous, correlated variables, X_1 and X_2, drawn from a bivariate normal distribution with mean 0, variance 1, and covariance -0.25; a continuous variable A^* drawn from a normal distribution with mean 1 and variance 1, and a binary variable D^* drawn from a Bernoulli distribution with the logit probability of success equal to $-3 + 0.5A^*$. Recall A^*, D^* were error-prone versions of the actual variables of interest, A, D. A was drawn from a Normal distribution with mean equal to $-X_1 + 0.5X_2 + 0.9A^* + 0.5D^*$ and variance 2. Finally, D was drawn from a Bernoulli distribution with the logit probability of success equal to $-5.5 - 2X_1 + X_2 + 5D^* + 0.5A$. Note that we assumed that there were no data errors for the other two variables, X_1 and X_2. Note also that we sampled A and D conditional on their error-prone counterparts, A^* and D^*, for ease of properly specifying the model.

```
set.seed(455)

#Load relevant packages
        require(mvtnorm)

#Fixed values
        obs<-4000
        ID<-rep(1:obs)
        X1X2cov<-(-0.25)
        X1var<-1
        X2var<-1
        Astar.var<-1
        A.var<-1
        beta.star<-c(1,0,0)
        gamma.star<-c(-3,0,0,0.5)
        beta<-c(0,-1,0.5,.9,0.5)
        gamma<-c(-5.5,-2,1,0.0,5.0,0.5)
        Nimpute<-50 #Number of imputations
```

```
#Generate values
#Covariates X and X2
        X1X2<-rmvnorm(obs, mean=c(0,0),sigma=matrix(c(X1var,X1X2cov,X1X2cov,X2var),2,2))
        X1<-X1X2[,1]
        X2<-X1X2[,2]
        Xmat<-cbind(1,X1,X2)
        rownames(Xmat)<-ID
        colnames(Xmat)<-c("Intercept","X1","X2")
####################################
#Unvalidated data
####################################
#A*
        A.star<-rnorm(obs,Xmat%*%beta.star,Astar.var)
#D*
        LP.D.star<-Xmat%*%gamma.star[1:ncol(Xmat)] +
                             A.star*gamma.star[length(gamma.star)]
        p.D.star<-exp(LP.D.star)/(1+ exp(LP.D.star))
        #Indicator of event (D*) at time t
        D.star<-rbinom(obs,1,p.D.star)
####################################
#Validated data
####################################
#A
        LP.A<-Xmat%*%beta[1:ncol(Xmat)] + A.star*beta[ncol(Xmat)+1] +
                    D.star*beta[ncol(Xmat)+2]
        A<-rnorm(obs,LP.A,A.var)
#D
        LP.D<-Xmat%*%gamma[1:ncol(Xmat)]  + A.star*gamma[ncol(Xmat)+1]  +
                    D.star*gamma[ncol(Xmat)+2] +A*gamma[ncol(Xmat)+3]
        p.D<- exp(LP.D)/(1+ exp(LP.D))
        #Indicator of event (D) at time t
        D<-rbinom(obs,1,p.D)
```

In the simulation, we investigated different validation subsample sizes (n= 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000) as well as different inclusion thresholds ($A > \{-\infty$, -1, -0.5, 0, 0.5, 1, 1.5, 2, 2.5, 3$\}$).

The code below assigned a validation subsample size (n=1,000) and inclusion threshold ($A > 2$). This generated a subset of 1,000 subjects that were randomly selected to represent an audited cohort with (A, D) known; for the remaining 3,000 subjects, (A, D) were treated as missing. X_1 and X_2 were treated as known for all 4,000 subjects.

```r
subsampleN<-1000
inthresh<-2

#These values are fixed for a given simulation, but may vary across simulations
        #Specify the audit/subsample size
        n.sample<-subsampleN
        #Specify the inclusion threshold
        inclusionthreshold<-inthresh

#Randomly sample records to represent audited cohort
        ChooseSubset<-sample(1:obs,n.sample,replace=F)
        sampled<-as.numeric((1:obs)%in%ChooseSubset)

#Original dataset
        dat<-dat.original<-data.frame(ID,X1,X2,A.star,D.star,A,D)
        dat$Intercept<-1
        dat$A<-ifelse(sampled==0,NA,dat$A)
        dat$D<-ifelse(sampled==0,NA,dat$D)
```

Simulation example: R code for imputation

In this setting, we specified a joint imputation model for (A, D) by fitting a linear regression model for A conditional on A^*, D^*, X_1 and X_2 and a logistic regression model for D conditional on A^*, D^*, X_1, X_2, and A using the subset of 1,000 audited records. For each imputation, the imputer accounted for both parameter uncertainty and random noise in the imputations. We used 50 multiple imputation replications.

```
dat.imputed<-dat
dat.imputed$Imputed<-as.numeric(sampled==0)
#Fit linear regression model using complete observations
    modA<-glm(A~X1+X2+A.star+D.star,data=dat,family="gaussian",y=FALSE,model=FALSE)
    modD<-glm(D~X1+X2+A.star+D.star+A,data=dat,family="binomial",y=FALSE,model=FALSE)



#Function to impute values
ImputationFn<-function(mod,datty){
vars<-c("Intercept",all.vars(formula(mod)[-2]))
newdatmat<-datty[,vars]
        #Account for parameter uncertainty
        #Re-draw from a multivariate normal distribution with mean and variance
         #as the parameter estimates and cov matrix from the regression model.
        desmatrix<-rmvnorm(1, mean=mod$coefficients, sigma=vcov(mod))
        linpred<-as.vector(desmatrix%*%t(newdatmat))

        if (mod$family[1]=="binomial"){ return(linpred)}
        if (mod$family[1]=="gaussian"){
        #Account for random noise
        #Randomly sample from the residuals
                resid.y<-mod$residuals
                imputed.values<-linpred +
                         as.numeric(sample(resid.y,length(linpred),replace=TRUE))
        }
        return(imputed.values)
}

        #Replace missing balues with imputed value
        imputed.A<-ImputationFn(modA,datty=dat.imputed)
        dat.imputed$A[sampled==0]<-imputed.A[sampled==0]
        imputed.LP.D<-ImputationFn(modD,datty=dat.imputed)
        imputed.D<-rbinom(length(imputed.LP.D),1,
                                exp(imputed.LP.D)/(1+exp(imputed.LP.D)))
        dat.imputed$D[sampled==0]<-imputed.D[sampled==0]
```

Simulation example: R code for analysis modeling

Our goal was to estimate the association between a predictor variable A and a binary outcome D among the subset of subjects with $A > 2$. Using the imputed dataset, we first excluded subjects with $A \leq 2$; we referred to this reduced dataset as the analysis dataset. Using this analysis dataset, a logistic regression analysis model was fit to estimate the association between A and D. Since the imputations were generated using observations from all subjects while the analysis model was based on just those with an imputed value of $A > 2$, there was now incompatibility between the imputation model and the analysis model.

```
#Generate dataset where obs meet inclusion criteria
dat.analysis<-dat.imputed[dat.imputed$A>inclusionthreshold,]
mod.analysis<-glm(D~A,data=dat.analysis,family="binomial")
analy.vars<-names(mod.analysis$coef)
analy.vars<-ifelse(analy.vars=="(Intercept)","Intercept",analy.vars)
```

Simulation example: R code for RW component calculations based on imputation model

The imputer needs to calculate and supply two datasets based on the score function of the imputation model, S_{mis} and d. The following code contains several functions. The first two functions evaluate the score function or its derivative for each subject. We then expand the function that was provided in the first example to calculate and output the two components (S_{mis} and d) for the RW variance estimator from the imputation model. Here, we allow for a joint imputation model and the variable being imputed can be binary or continuous. This function is intended to be used for each imputation.

```
#Function to evaluate score function
#Code here works when imputation model is linear or logistic regression
S_u_function<-function(datty,mod.impute,imputedvar){
        imputemod.vars<-c("Intercept",all.vars(formula(mod.impute)[-2]))

if (mod.impute$family[1]=="gaussian"){
        sigma.est<-var(mod.impute$residuals)
        pred.values<-predict(mod.impute,newdata=datty)
        S_u_1<-(imputedvar-pred.values)*datty[,imputemod.vars]/sigma.est
        S_sigma_1<-0.5*(-1/sigma.est+ (imputedvar-pred.values)^2/sigma.est^2)
        S_u<-cbind(S_u_1,S_sigma_1)
        modelvarcount<-length(imputemod.vars)+1 #Account for sigma
}

if (mod.impute$family[1]=="binomial"){
        #pred.values is actually predicted LP
        pred.values<-predict(mod.impute,newdata=datty,returnvar="linpred")
        S_u<-datty[,imputemod.vars]*
                        (imputedvar-exp(pred.values)/(1+exp(pred.values)))
        modelvarcount<-length(imputemod.vars)
}
return(S_u)
}

#Function to evaluate derivative of score function
#Code here works when imputation model is linear or logistic regression
S2_function<-function(datty,mod.impute,imputedvar){
    imputemod.vars<-c("Intercept",all.vars(formula(mod.impute)[-2]))
    impvar_n<-length(imputemod.vars)

if (mod.impute$family[1]=="gaussian"){
        sigma.est<-var(mod.impute$residuals)
        pred.values<-predict(mod.impute,newdata=datty)

        S2_uu<-matrix(NA,nrow(datty),length(imputemod.vars))
        S2_uu<-with(datty,(-1/sigma.est*t(datty[Imputed==0,c(imputemod.vars)])%*%
                t(t(datty[Imputed==0,c(imputemod.vars)]))))
        S2_usigma<-with(datty,(-1/sigma.est^2*t(datty[Imputed==0
                ,c(imputemod.vars)])%*%(imputedvar-pred.values)[Imputed==0]))
        S2_sigmasigma<-sum(with(datty,ifelse(Imputed==1,0,
                1/(2*sigma.est^2)-(imputedvar-pred.values)^2/sigma.est^3)))
        S2_mod<-matrix(0,impvar_n+1,impvar_n+1)
        S2_mod[1:impvar_n,1:impvar_n]<-S2_uu/nrow(datty)
        S2_mod[1:impvar_n,impvar_n+1]<-apply(S2_usigma,1,mean)/nrow(datty)
        S2_mod[impvar_n+1,1:impvar_n]<-apply(S2_usigma,1,mean)/nrow(datty)
        S2_mod[impvar_n+1,impvar_n+1]<-S2_sigmasigma/nrow(datty)
}
if (mod.impute$family[1]=="binomial"){
        ImputeX<-datty[,imputemod.vars]
        #pred.values is actually predicted LP
        pred.values<-predict(mod.impute,newdata=datty,type="link")
        S2_mod<-matrix(NA,ncol(ImputeX),ncol(ImputeX))
        constant.term<-exp(pred.values)/(1+exp(pred.values))^2
        for (matrow in 1:ncol(ImputeX)){
                for (matcol in 1:ncol(ImputeX)){
                temp<-(-1)*(ImputeX[,matrow])*((ImputeX[,matcol]))*constant.term
                temp[datty$Imputed==1]<-0
                S2_mod[matrow,matcol]<-sum(temp)/length(datty$ID)
          }
        }
}
  return(S2_mod)
}

#Function that calculates and outputs the required components from imputation model
#Updated from Example 1 to allow for joint imputation models
RW_Components_Imputation_Fn<-function(){

        #Evaluate score function with respect to all parameters in the models
        S_u_mod1<-S_u_function(dat.imputed,modA,dat.imputed[,rownames(attributes(modA$terms)$factors)[1]])
        S_u_mod2<-S_u_function(dat.imputed,modD,dat.imputed[,rownames(attributes(modD$terms)$factors)[1]])

        S_u<-data.frame(S_u_mod1,S_u_mod2)

        #Preliminary matrix that indicates missingess for each subject
        ImputedMat<-matrix(dat.imputed$Imputed==1,nrow(S_u),ncol(S_u),byrow=FALSE)
        NOTImputedMat<-1-ImputedMat

        ################################################################
        #Calculation of S_mis -- component from imputation model
        #Output a dataset that is the evaluated score function for imputed obs
        ################################################################
        S_mis_imp<-S_u*ImputedMat

        ################################################################
        #Calculation of D -- component from imputation model
        ################################################################
        S_orig<-S_u*NOTImputedMat
        S2_mod1<-S2_function(dat.imputed,modA,dat.imputed[,rownames(attributes(modA$terms)$factors)[1]])
        S2_mod2<-S2_function(dat.imputed,modD,dat.imputed[,rownames(attributes(modD$terms)$factors)[1]])
        n_S2mod1<-ncol(S2_mod1)
        n_S2mod2<-ncol(S2_mod2)
        n_S2mod<-n_S2mod1+n_S2mod2

        S2<-matrix(0,n_S2mod,n_S2mod)
        S2[1:n_S2mod1,1:n_S2mod1]<-S2_mod1
        S2[(n_S2mod1+1):n_S2mod,(n_S2mod1+1):n_S2mod]<-S2_mod2

        Dmat<-solve(S2)
        d_t<-((-1)*Dmat)%*%t(S_orig)
        d<-t(d_t)
        return(list(S_mis_imp,d))
}
```

Simulation example: R code for RW component calculations based on analysis model

The analyst needs to calculate and supply two datasets, u and τ. The following code contains a function that will calculate and output u and τ for a logistic regression analysis model. Note that our analysis model for this example (logistic regression) is different than the analysis model in the first example (linear regression). As a result, the code to calculate u and τ has been changed. These calculations are performed for each imputation.

```
RW_Components_Analysis_Fn<-function(){
        analysis.predict<-predict(mod.analysis,type="response")

        ##############################################################
        #Calculation of u
        #Evaluate the estimating equation for each observation
        ##############################################################
        U_imp_pre<-dat.analysis[,analy.vars]*
                                matrix((dat.analysis[,rownames(attributes(mod.analysis$terms)$factors)[1]]
                                                -analysis.predict), dim(dat.analysis[, analy.vars]))
        aa<-cbind(dat.analysis$ID,U_imp_pre)
        colnames(aa)<-c("ID",colnames(U_imp_pre))
        bb<-merge(dat.imputed[,c("ID","Imputed")],aa,by="ID",all=T)
        U_imp<-bb[,colnames(U_imp_pre)]
        U_imp[is.na(U_imp)]<-0

        ##############################################################
        #Calculation of tau
        #Take the derivative of the estimating equation and evaluate for each obs
        ##############################################################
        tempdat<-dat.analysis[,analy.vars]
        #pred.values is actually predicted LP
        analysis.predLP<-predict(mod.analysis,newdata=tempdat,type="link")
        tau_imp<-matrix(NA,ncol(tempdat),ncol(tempdat))
        rownames(tau_imp)<-colnames(tau_imp)<-colnames(tempdat)
        constant.term<-exp(analysis.predLP)/(1+exp(analysis.predLP))^2
        for (matrow in 1:ncol(tempdat)){
                for (matcol in 1:ncol(tempdat)){
```

```
                temp<-(-1)*(tempdat[,matrow])*((tempdat[,matcol]))*constant.term

                tau_imp[matrow,matcol]<-(-1)*sum(temp)

           }

        }

        return(list(U_imp,tau_imp))

}
```

Simulation example: R code for RW multiple imputation variance calculation

The above components were calculated using one imputed dataset. The following code provides an outline for calculating the variance estimator across multiple imputations. For this example, the dataset was imputed 50 times. We output the variance estimates for the RW variance estimator and RR variance estimator for comparison.

```
#Initialize select variables

    U_imp_sum<-0

    kappa_sum<-0

    tau_sum<-0

    analysis.est<-analysis.se<-analysisN<-NULL

for (p in 1:Nimpute){

        #Replace missing values with imputed value

        dat.imputed<-dat

        dat.imputed$Imputed<-as.numeric(sampled==0)


        imputed.A<-ImputationFn(modA,datty=dat.imputed)

        dat.imputed$A[sampled==0]<-imputed.A[sampled==0]

        imputed.LP.D<-ImputationFn(modD,datty=dat.imputed)

        imputed.D<-rbinom(length(imputed.LP.D),1,exp(imputed.LP.D)/

                (1+exp(imputed.LP.D)))

        dat.imputed$D[sampled==0]<-imputed.D[sampled==0]


        #Create analysis dataset based on inclusion criteria

        #Perform logistic regression

        dat.analysis<-dat.imputed[dat.imputed$A>inclusionthreshold,]

        mod.analysis<-glm(D~A,data=dat.analysis,family="binomial")

        analy.vars<-names(mod.analysis$coef)

        analy.vars<-ifelse(analy.vars=="(Intercept)","Intercept",analy.vars)


        #Capture estimates and corresponding SE from each imputation

        analysis.est[p]<-mod.analysis$coef["A"]
        analysis.se[p]<-sqrt(diag(vcov(mod.analysis))["A"])

        #Evaluate all four components for a given imputation
        ImputationComponents<-RW_Components_Imputation_Fn()
        AnalysisComponents<-RW_Components_Analysis_Fn()
        S_mis_imp<-ImputationComponents[[1]]
        d<-ImputationComponents[[2]] #Only need to calculate d once
        U_imp<-AnalysisComponents[[1]]
        U_imp[is.na(U_imp)]<-0
        tau_imp<-AnalysisComponents[[2]]

        #Create summations of certain components across imputations
        U_imp_sum<-U_imp_sum+U_imp
        tau_sum<-tau_sum+tau_imp

        #Calculate kappa for a given imputation
        kappa_imp<-t(U_imp)%*%t(t(S_mis_imp))
        #Create summation for kappa across imputations
        kappa_sum<-kappa_sum+kappa_imp

        #Save the number of observations in the analysis dataset
        analysisN[p]<-nrow(dat.analysis)
}
############################################################### #
#Combine components together to calculate Robins and Wang variance estimator
############################################################### #
        u_bar<-U_imp_sum/Nimpute
        u_bar<-t(t(u_bar))

        omega<-(t(u_bar)%*%t(t(u_bar)))/(obs)
        kappa<-t(t(kappa_sum/(obs*Nimpute)))

        alpha<-(t(d)%*%d)/obs

        delta<-omega + kappa%*%alpha%*%t(kappa) +
                (1/obs)*(kappa%*%t(d)%*%u_bar + t(kappa%*%t(d)%*%u_bar))
        tau<-tau_sum/(Nimpute*obs)
        GAMMA<-(1/obs)*t(t(solve(tau)))%*%delta%*%t(solve(tau))

        #Save SE estimates to calculate CIs!
        RobinsWangSE<-sqrt(diag(GAMMA)["A"])
        RubinSE<-sqrt( mean(analysis.se^2)+ (Nimpute+1)/Nimpute*var(analysis.est))
        #Output imputation estimate for parameter
        mean(analysis.est)

## [1] 0.8928581

        #Output variance estimates for RW and Rubin
        RobinsWangSE^2

##           A
## 0.005834417

        RubinSE^2

## [1] 0.01020951
```

Simulation example: Results

The imputation estimate for the coefficient was 0.8929 with corresponding variance using Robins and Wang's approach, 0.0058. For comparison, the estimated variance based on Rubin's approach was 0.0102. Across 50 imputations, the average analysis dataset size was 1145 after excluding subjects with $A \leq 2$. The smallest and largest analysis datasets were 1074 and 1200 subjects, respectively.

Web Appendix 3

The statistical analyses corresponding to the motivating example in the manuscript was performed using the code provided below. Please note that functions used to calculate various Robins and Wang components for the simulation study (Web Appendix 2) are also used for this example.

Motivating example: R code for data import

The following R code imports a dataset containing from a cohort of 3,526 HIV patients who established care and enrolled at least 12 months before the study freeze date.

```
        set.seed(45)

#Import necessary data
#Note: this data will not be available to external audience
        load("R datasets/datRW.Rda")


#Load necessary libraries
        require(rms)
        detach(package:rms)
        detach(package:Hmisc)
        require(plyr)
        require(mvtnorm)
        require(MASS)
        require(data.table)
        require(rms)


#Fixed values
        Nimpute<-50
        n<-length(unique(datRW$CFAR_PID))
        obs<-n
```

```r
#Generate a few useful variables
        datRW$ID<-datRW$CFAR_PID
        datRW$Intercept<-1
```

We then randomly selected 1,000 records (28.4%) as having been validated; validated data were masked for the remaining 2,526 records.

```r
        n.sample<-1000



#Randomly sample records to represent audited cohort
        dq<-datRW
        AllIDs<-unique(dq$CFAR_PID)
        ChooseSubset<-as.character(sample(AllIDs,n.sample,replace=FALSE))
        dq$sampled<-ifelse(dq$CFAR_PID %in% ChooseSubset,1,0)
```

Motivating example: R code for imputation

Using both unvalidated and validated data for the 1,000 validated patient records, we constructed an imputation model for the joint distribution of the true values of the two error-prone variables (any ART dispensation in the first month and any ADE within 12 months) conditional on the unvalidated values of these same variables and other covariates. Specifically, we fit a logistic regression model for any ART dispensation conditional on the unvalidated ART dispensation variable, the unvalidated ADE variable, and other covariates. We also fit a logistic regression model for ADE within 12 months conditional on the unvalidated ADE variable, the unvalidated ART dispensation variable, the validated ART dispensation variable, and other covariates. In both models, the other covariates were calendar year of enrollment, baseline CD4 count, and baseline log viral load.

```r
#Prepare datasets for imputation model
        alldata<-dq
        alldata$Dispensation.v[alldata$sampled==0]<-NA
        alldata$OI.v[alldata$sampled==0]<-NA



        #Specify variables used in analysis (both spline and non-spline versions)
        varlist.for.art.splines<-c("Dispensation.nv","rcs(CD4,3)","rcs(logVL,3)","OI.nv",
                                    "rcs(YEAR_OF_ENROLLMENT,3)")
```

```
varlist.for.art.nosplines<-c("Dispensation.nv","CD4","logVL","OI.nv","YEAR_OF_ENROLLMENT")
varlist.for.art.splines<-varlist.for.art.nosplines[1:6]

#Need 4 knot points for logVL as 3 gives warning message
varlist.for.ois.splines<-c("OI.nv","Dispensation.v","Dispensation.nv","rcs(CD4,3)","rcs(logVL,4)",
                                      "rcs(YEAR_OF_ENROLLMENT,3)")
varlist.for.ois.nosplines<-c("OI.nv","Dispensation.v","Dispensation.nv","CD4","logVL","YEAR_OF_ENROLLMENT")
varlist.for.ois.splines<-varlist.for.ois.nosplines[1:7]



#Set values to null/zero
modfit.logistic<-0
mod.art<-mod.oi<-NULL
data.OIinclude<-NULL
my.formula.art<-my.formula.oi<-NULL

##################################
#Imputation model for ART
##################################
#ART - anytime
#Assign formula, run regression
my.formula.art <- as.formula(paste( 'Dispensation.v', '~', paste(varlist.for.art.splines, collapse='+' )))
if(!is.null(tryCatch(mod.art<-glm(my.formula.art,data=alldata,family="binomial",maxit=200),
                                  error = function(e) {}))) { modfit.logistic<-         modfit.logistic+1}
#If error message (usually, rank deficiency) we will rerun without splines
counter<-0
while(is.null(mod.art)){
    my.formula.art <- as.formula(paste( 'Dispensation.v', '~',
                 paste(varlist.for.art.nosplines[-(length(varlist.for.art.nosplines)-0:counter)],
                 collapse=' + ' )))
            if(!is.null(tryCatch(mod.art<-glm(my.formula.art ,data=alldata,family="binomial",maxit=200) ,
                                   error = function(e) {}))) { modfit.logistic<- modfit.logistic+1}
counter<-counter+1
}



##################################
#Imputation model for OIs
##################################
#Models for OIs -- complicated b/c we may remove certain OIs due to small sample sizes!
OIlist<-c("CYTO.OI", "DEMENTIAENCEPHALOPATHY.OI", "ESOPHAGEALorCANDIDIASIS.OI",
               "HISTOPLASMOSIS.OI", "KS.OI",  "LYMPHOMA.OI", "MAC.OI", "MENINGITIS.OI", "PCP.OI",
                "PNEUMONIA.OI", "RETINOPHANYorRETINITIS.OI", "TB.OI", "WASTING.OI", "WEIGHT.OI")
#OI - anytime
data.OIcount<-sort(apply(alldata[alldata$sampled==1,OIlist],2,sum),decreasing=T)
data.OIinclude<-names(data.OIcount[data.OIcount>30])  #Remove OIs with less than 5 observations!
if (length(data.OIinclude)==0) data.OIinclude<-1  #Just in case no OIs >4!
my.formula.oi <- as.formula(paste( 'OI.v', '~', paste(varlist.for.ois.splines, collapse=' + ' ), '+',
                               paste(data.OIinclude, collapse=' + ' ) ))
if(!is.null(tryCatch(mod.oi<-glm(my.formula.oi ,data=alldata,family="binomial",maxit=200,
                            y=FALSE, model=FALSE) ,
                            error = function(e) {}))) { modfit.logistic<- modfit.logistic+1}

#If model does not work, re-run without splines AND start removing OIs
counter<-0
while(is.null(mod.oi)){
    #Iteratively remove variables under model works
    newvarlist<-c(varlist.for.ois.nosplines[1:length(varlist.for.ois.nosplines)],data.OIinclude)
    my.formula.oi <- as.formula(paste( 'OI.v', '~', paste(newvarlist[-(length(newvarlist)-0:counter)],
                                                 collapse=' + ' )))
    if(!is.null(tryCatch(mod.oi<-glm(my.formula.oi ,data=alldata,family="binomial",maxit=200,
                            y=FALSE,model=FALSE) ,
                            error = function(e) {}))) { modfit.logistic<- modfit.logistic+1}
    counter<-counter+1
 }




#Give generic names to imputation model outputs
modfit<-modfit.logistic
mod.selected11<-mod.art
mod.selected21<-mod.oi



#Create datasets that add in imputed values for key values for records that were not sampled
#Set up the backbone of those datasets prior to the imputation loop
#Some continuous variables were modeled as splines in imputation models
#Identify those variables based on the regression model
#Transform them in the spline components based upon chosen knot locations

splinedatfn<-function(analysisdat,varlist,varlistnosplines,mod,outcomevar){
        yyy<-model.frame(mod)
        summ.fit<-summary(mod)
        analysisdat.splines<-rep(1,dim(analysisdat)[1])
        for (i in 2:length(yyy)){
          if (is.null(attributes(yyy[,i]))) {
              if (colnames(yyy)[i] %in%  rownames(summ.fit$coef)) {
                            analysisdat.splines<-cbind( analysisdat.splines,analysisdat[,colnames(yyy)[i]])
                        }
          }
          if (is.null(attributes(yyy[,i]))==F) {
              attrib<-attributes(yyy[,i])
              if (attrib$assume=="rcspline")
              {
                  analysisdat.splines<-cbind(analysisdat.splines,analysisdat[,attrib$name],
                                  rcspline.eval(analysisdat[,attrib$name],knots=attrib$parms))
              }
          }
        }
        analysisdat.splines<-data.frame(as.character(analysisdat$CFAR_PID),
                                         analysisdat$sampled,analysisdat[,outcomevar],
                                        analysisdat.splines,stringsAsFactors=FALSE)
        colnames(analysisdat.splines)<-c("ID","sampled",outcomevar,rownames(summ.fit$coef))
        # analysisdat.splines<-data.frame(analysisdat.splines)
        return(analysisdat.splines)
}



dat.splines.art<-splinedatfn(analysisdat=dq, varlist.for.art.splines, varlist.for.art.nosplines,
                                            mod=mod.selected11, outcomevar="Dispensation.v")
dat.splines.art<-dat.splines.art[order(dat.splines.art$ID),]
tempdat.art<-as.matrix(dat.splines.art[,4:ncol(dat.splines.art)])

####################################################################################
####################################################################################
varlist.for.ois.splines2<-names(attr(mod.selected21$terms,"dataClasses"))[-1]
varlist.for.ois.splines2<-gsub(" ","",varlist.for.ois.splines2)

varlist.for.ois.nosplines2<-c(varlist.for.ois.nosplines,
                 varlist.for.ois.splines2[!varlist.for.ois.splines2%in%varlist.for.ois.splines])
dat.splines.oi<-splinedatfn(analysisdat=dq, varlist.for.ois.splines2, varlist.for.ois.nosplines2,
                                           mod=mod.selected21, outcomevar="OI.v")
dat.splines.oi<-dat.splines.oi[order(dat.splines.oi$ID),]
```

```
        mod.selected11.summ<-summary(mod.selected11)
        mod.selected21.summ<-summary(mod.selected21)
```

Motivating example: R code for analysis modeling

Our goal was to estimate the association between CD4 cell count at enrollment (baseline) and subsequent outcomes during the first year among patients that initiated antiretroviral therapy (ART) at enrollment. A patient was classified as having a poor outcome if they died, had an AIDS-defining event (ADE), or were lost to follow-up during the first 12 months after enrollment. For this study, patients were excluded from analyses if they did not have an ART dispensation within the first month after enrollment, if they did not establish care at the clinic (defined as having less than three months of follow-up), or if they enrolled at the clinic less than 12 months before the data freeze date.

Using the imputed dataset, we first excluded subjects that did not have an ART dispensation in the first month; we referred to this reduced dataset as the analysis dataset. Using this analysis dataset, a logistic regression analysis model was fit to estimate the association.

Since the imputations were generated using observations from all subjects while the analysis model was based on just those with an ART dispensation in the first month, there was now incompatibility between the imputation model and the analysis model. To account for this incompatibility when estimating the imputation variance, we will implement the Robins and Wang approach. The code below uses the same functions used in the simulation example to calculate the Robins and Wang components; see above.

```
                #Useful functions
                 ##################################################################################
                #Re-draw new estimated coeff using a MVN distribution
                #mean = regression model estimates
                #variance = covariance matrix from regression model
               ##################################################################################
                desmatrixfn<-function(mod,summ.mod,Nobs=Nimpute){
                        desmatrix<-rmvnorm(Nobs, mean=mod$coeff[rownames(summ.mod$coef)],
                                           sigma=(summ.mod$dispersion*summ.mod$cov.unscaled[rownames(summ.mod$coef),
                                           rownames(summ.mod$coef)]))
                }

                 ##################################################################################
                #Adds noise to imputation estimates
                 ##################################################################################
                ParmNoise<-function(fit,summ.fit,newdatmat,returnvar="pred"){
                    sigmas<-NULL
                    desmatrix<-NULL
                    desmatrix<-desmatrixfn(fit,summ.fit,Nobs=1)
                    linpred<-as.vector(desmatrix%*%t(newdatmat))
                    #When our model does a very good job of predicting outcome, can get large linear predictors
                    #When taking inverse logit below, R gives us a value of 0 (instead of 1)
                    #By artificially truncating these values, we can ensure the code runs as intended
                        linpred<-ifelse(linpred>100,100,linpred)

                        if (fit$family[1]=="gaussian"){
                          resid.y<-fit$residuals
                          linpred<-linpred + as.numeric(sample(resid.y,length(linpred),replace=TRUE))
                      }

                        if (returnvar=="linpred") return(linpred)
                        if (returnvar=="pred") {
                          pred<-rbinom(length(linpred),1,exp(linpred)/(1+exp(linpred)))
                          return(pred)
                        }
                    }


    #Initialize select variables
      U_imp_nonzeros<-0
      U_imp_sum<-0
      kappa_sum<-0
      tau_sum<-0
      omega_sum<-0
      analysis.est<-analysis.se<-analysis.nonrobust.se<-analysisN<-NULL
    for (p in 1:Nimpute){
        modA<-mod.selected11
        modD<-mod.selected21

        ###############################################################################################
        #ART
        ###############################################################################################
        ART.predict.linpred<-ParmNoise(mod.selected11, mod.selected11.summ,
                                                    tempdat.art, returnvar="linpred")
        predicted.LP.ART<-ART.predict.linpred
        pred.ART<-rbinom(length(ART.predict.linpred),1,
                               exp(ART.predict.linpred)/(1+exp(ART.predict.linpred)))
        dat.splines.art$Dispensation.v[dat.splines.art$sampled==0]<-
```

```
                                                    as.vector(pred.ART)[dat.splines.art$sampled==0]
###############################################################################################
##OI
###############################################################################################
#Need to update dispensation values here in order to properly impute OI status
dat.splines.oi$Dispensation.v[dat.splines.oi$sampled==0]<-
                                         as.vector(pred.ART)[dat.splines.art$sampled==0]
tempdat.oi<-as.matrix(dat.splines.oi[,4:ncol(dat.splines.oi)])
predicted.LP.OI<-OI.predict.linpred<-ParmNoise(mod.selected21,mod.selected21.summ,tempdat.oi,
                                                                          returnvar="linpred")
pred.OI<-rbinom(length(OI.predict.linpred),1,exp(OI.predict.linpred)/(1+exp(OI.predict.linpred)))
dat.splines.oi$OI.v[dat.splines.oi$sampled==0]<-as.vector(pred.OI)[dat.splines.oi$sampled==0]



###############################################################################################
######################    Generate Analysis Dataset       ##########################
###############################################################################################

aaa<-data.table(data.frame(dq[,c("CFAR_PID","sampled","OI.v")],pred.OI),key=c("CFAR_PID"))
bbb<-data.table(data.frame(dq[,c("CFAR_PID","Dispensation.v")],pred.ART),key=c("CFAR_PID"))
dat<-data.frame(bbb[aaa,])

dat.imputed<-datRW
dat.imputed$Dispensation.v<-with(dat,ifelse(sampled==1,Dispensation.v,pred.ART))
dat.imputed$OI.v<-with(dat,ifelse(sampled==1,OI.v,pred.OI))
dat.imputed$Imputed<-as.numeric(dat$sampled==0)




dat.analysis<-dat.imputed[dat.imputed$Dispensation.v==1,]
mod.analysis<-glm(OI.v~ CD4 + logVL + YEAR_OF_ENROLLMENT ,data=dat.analysis,family="binomial")
mod.analysis.robust<-robcov(lrm(OI.v~ CD4 + logVL + YEAR_OF_ENROLLMENT ,
                                                        data=dat.analysis,y=TRUE,x=TRUE))
analy.vars<-names(mod.analysis$coef)
analy.vars<-ifelse(analy.vars=="(Intercept)","Intercept",analy.vars)



#Capture estimates and corresponding SE from each imputation
analysis.est[p]<-mod.analysis$coef["CD4"]
analysis.nonrobust.se[p]<-sqrt(diag(vcov(mod.analysis))["CD4"])
analysis.se[p]<-sqrt(diag(mod.analysis.robust$var)["CD4"])
```

```r
        #Evaluate all four components for a given imputation
        ImputationComponents<-RW_Components_Imputation_Fn()
        AnalysisComponents<-RW_Components_Analysis_Fn()
        S_mis_imp<-ImputationComponents[[1]]
        d<-ImputationComponents[[2]] #Only need to calculate d once as it does not vary across imputations
        U_imp<-AnalysisComponents[[1]]
        U_imp[is.na(U_imp)]<-0
        tau_imp<-AnalysisComponents[[2]]

        #04DEC2018 addition - how to better handle varying inclusion across imputations for omega
        U_imp_nonzeros<-(dat.imputed$ID%in%dat.analysis$ID)*1+U_imp_nonzeros
        omega_imp<-t(U_imp)%*%t(t(U_imp))
        omega_sum<-omega_sum+omega_imp

        #Create summations of certain components across imputations
        U_imp_sum<-U_imp_sum+U_imp
        tau_sum<-tau_sum+tau_imp
        #Calculate kappa for a given imputation
        kappa_imp<-t(U_imp)%*%t(t(S_mis_imp))
        #Create summation for kappa across imputations
        kappa_sum<-kappa_sum+kappa_imp

        #Save the number of observations in the analysis dataset
        analysisN[p]<-nrow(dat.analysis)
 }
#End imputation loop

 ############################################################### #
 #Combine components together to calculate Robins and Wang variance estimator
 ############################################################### #
        u_bar<-U_imp_sum/Nimpute
        u_bar<-t(t(u_bar))

 # 04DEC2018 - explore a few extra changes for omega
        #omega<-omega_sum/(obs*Nimpute)
        omega<-(t(u_bar)%*%t(t(u_bar)))/(obs)



        #Not taking averages!
        omega_sum_alt<-omega*0
         #Make it so it is average across instances where actually observed
        u_bar_temp<-t(U_imp_sum/matrix(U_imp_nonzeros,nrow(U_imp_sum),ncol(U_imp_sum),byrow=FALSE))
```

```
                u_bar_temp[is.na(u_bar_temp)]<-0
                for (rr in 1:nrow(u_bar_temp)){
                        for (qq in 1:nrow(u_bar_temp)){
                            omega_sum_alt[rr,qq]<-sum(u_bar_temp[rr,]*u_bar_temp[qq,]*U_imp_nonzeros/Nimpute)
                        }
                }
                omega<-omega_sum_alt/(obs)

                kappa<-t(t(kappa_sum/(obs*Nimpute)))

                alpha<-(t(d)%*%d)/obs

                delta<-omega + kappa%*%alpha%*%t(kappa) +
                        (1/obs)*(kappa%*%t(d)%*%u_bar + t(kappa%*%t(d)%*%u_bar))
                tau<-tau_sum/(Nimpute*obs)
                GAMMA<-(1/obs)*t(t(solve(tau)))%*%delta%*%t(solve(tau))

                #Save SE estimates to calculate CIs!
                RobinsWangSE<-sqrt(diag(GAMMA)["CD4"])
                RubinSE<-sqrt( mean(analysis.se^2)+ (Nimpute+1)/Nimpute*var(analysis.est))
                RubinNonRobustSE<-sqrt( mean(analysis.nonrobust.se^2)+ (Nimpute+1)/Nimpute*var(analysis.est))
                #Output imputation estimate for parameter
                Est<-mean(analysis.est)
                 Est*100

## [1] -0.1132526

                #Output variance estimates for RW and Rubin
                RobinsWangSE^2 *100

##          CD4
## 2.139962e-05

                RubinSE^2  *100

## [1] 4.152205e-05

                RubinNonRobustSE^2 *100

## [1] 4.203021e-05
```

Motivating Example: Results

The estimated log odds of a poor outcome was 0.113 lower for every 100 unit increase in CD4 count. The corresponding imputation variance estimate for the log odds was 29% smaller using RW (0.046) relative to RR (0.065). When calculating confidence intervals corresponding to the odds ratio (OR = 0.89), the RW-based 95% CI (0.82, 0.98) was narrower than the RR-based 95% CI (0.79, 1.01).