eAppendix for: “An Improvement in estimation of the standard error for the natural direct effect in causal mediation analysis”

Xin Gao and Li Luo*

1. Drawbacks of the estimator for $\text{Var}(MSE)$ in the previous literature and the corresponding proofs

Valeri and VanderWeele used $S_k = MSE$ to estimate $\Sigma_{\sigma^2} = \text{Var}(MSE)$ in their SAS macro for causal mediation analysis with survival data for inference of natural direct effect (Valeri & VanderWeele, 2015; Valeri & VanderWeele, 2013). We noted that this estimator mainly has two drawbacks. First, S_k is a biased estimator in most cases with any fixed $k = n - p - 1$ and σ^2 . Second, S_k converges in probability to σ^2 as well as Σ_{σ^2} converges everywhere (also in probability) to 0 with any fixed σ^2 , as k goes to infinity. The proofs for these two drawbacks are provided below.

Theorem 1. Let $E(M|A, C) = \beta_0 + \beta_1 a + \beta_2' c$ be the mediator linear regression model, where M, A and C represent a continuous mediator, an exposure and additional covariates, respectively. Let n be the number of observations, p be the number of predictors, $k = n - p - 1$ and MSE be the mean squared error in the sense of linear regression. Let SSE be the sum of squared errors and σ^2 be the variance of the random error, both of which are from the same mediator linear regression model. Then, S_k is a biased estimator for Σ_{σ^2} in most cases with any fixed k and σ^2 .

Proof: First, we need to find Σ_{σ^2} by using the facts that $MSE = \frac{SSE}{n-p-1} = \frac{SSE}{k}$ and $\frac{SSE}{\sigma^2} \sim \chi^2_{(k)}$.

$$\begin{aligned}\Sigma_{\sigma^2} &= \text{Var}(MSE) \\ &= \text{Var}\left(\frac{SSE}{k}\right) \\ &= \text{Var}\left(\frac{SSE}{\sigma^2} \frac{\sigma^2}{k}\right) \\ &= \frac{\sigma^4}{k^2} \text{Var}\left(\frac{SSE}{\sigma^2}\right) \\ &= \frac{2\sigma^4}{k^2} k \\ &= \frac{2\sigma^4}{k}\end{aligned}$$

The expected value of $S_k = MSE$ is σ^2 (Christensen, 2011). In order for S_k to be an unbiased estimator, the following equation must be true.

$$\begin{aligned}\sigma^2 &= \frac{2\sigma^4}{k} \\ \Leftrightarrow \\ \sigma^2 &= \frac{k}{2}\end{aligned}$$

However, this condition can hardly be met, and therefore S_k is a biased estimator in most cases.
□

Theorem 2. S_k converges in probability to σ^2 as well as Σ_{σ^2} converges everywhere (also in probability) to 0 with any fixed σ^2 , as k goes to infinity under the conditions described in **Theorem 1**.

Proof: First, we need to show S_k converges in probability to σ^2 with any fixed σ^2 as k goes to infinity. Since S_k is unbiased for σ^2 , we only need to consider its variance.

$$\begin{aligned}\lim_{k \rightarrow \infty} \text{Var}(S_k) &= \lim_{k \rightarrow \infty} \frac{2\sigma^4}{k} \\ &= 0\end{aligned}$$

Hence, S_k is a consistent sequence of estimators of σ^2 , and namely, S_k converges in probability to σ^2 (Casella & Berger, 2002).

Next, we need to show Σ_{σ^2} converges everywhere (also in probability) to 0 with any fixed σ^2 , as k goes to infinity.

$$\begin{aligned}\lim_{k \rightarrow \infty} \Sigma_{\sigma^2} &= \lim_{k \rightarrow \infty} \frac{2\sigma^4}{k} \\ &= 0\end{aligned}$$

Obviously, the convergence is everywhere (Rudin, 1953), and everywhere convergence implies convergence in probability (Rudin, 1953; Tao, 2011; Casella & Berger, 2002). Therefore, Σ_{σ^2} converges everywhere (also in probability) to 0. □

2. Properties of the new estimator and the corresponding proofs

We provide a new estimator $W_k = \frac{2MSE^2}{k+2}$ for $\Sigma_{\sigma^2} = \text{Var}(MSE)$ that possesses three desired properties and can overcome the drawbacks of the old one. First, W_k is an unbiased estimator with any fixed k and σ^2 . Second, W_k achieves the minimum variance among all unbiased estimators for Σ_{σ^2} with any fixed k and σ^2 . Third, W_k converges in probability to 0 with any fixed σ^2 , as k goes to infinity, which is the same limit as Σ_{σ^2} converges to.

Theorem 3. W_k is an unbiased estimator of Σ_{σ^2} with any fixed k and σ^2 under the condition described in **Theorem 1**.

Proof: We know from **Theorem 1** that $\Sigma_{\sigma^2} = \frac{2\sigma^4}{k}$. Therefore, we just need to show the expected value of W_k equals $\frac{2\sigma^4}{k}$.

$$\begin{aligned}
E(W_k) &= E\left(\frac{2MSE^2}{k+2}\right) \\
&= \frac{2}{k+2} E(MSE^2) \\
&= \frac{2}{k+2} (\text{Var}(MSE) + (E(MSE))^2) \\
&= \frac{2}{k+2} \left(\frac{2\sigma^4}{k} + \sigma^4\right) \\
&= \frac{2}{k+2} \frac{(k+2)\sigma^4}{k} \\
&= \frac{2\sigma^4}{k}
\end{aligned}$$

Therefore, W_k is an unbiased estimator for Σ_{σ^2} . $\square$

Theorem 4. W_k achieves the minimum variance among all unbiased estimators for Σ_{σ^2} with any fixed k and σ^2 under the condition described in **Theorem 1**.

Proof: Let $X = \frac{SSE}{\sigma^2} \sim \chi_{(k)}^2$ and $Y = MSE = \frac{\sigma^2}{k} X$. We have the following result by using univariate transformation.

$$\begin{aligned}
f_Y(y|\sigma^2) &= \frac{k}{\sigma^2} \frac{1}{\Gamma\left(\frac{k}{2}\right) 2^{\frac{k}{2}}} \left(\frac{ky}{\sigma^2}\right)^{\frac{k}{2}-1} \exp\left(-\frac{ky}{2\sigma^2}\right) \\
&= \left(\frac{k}{\sigma^2}\right)^{\frac{k}{2}} \frac{1}{\Gamma\left(\frac{k}{2}\right) 2^{\frac{k}{2}}} y^{\frac{k}{2}-1} \exp\left(-\frac{y}{\frac{2\sigma^2}{k}}\right) \\
&= \frac{1}{\Gamma\left(\frac{k}{2}\right) \left(\frac{2\sigma^2}{k}\right)^{\frac{k}{2}}} y^{\frac{k}{2}-1} \exp\left(-\frac{y}{\frac{2\sigma^2}{k}}\right)
\end{aligned}$$

This is a $\Gamma\left(\frac{k}{2}, \frac{2\sigma^2}{k}\right)$ distribution, where k is known, and therefore it belongs to an exponential family. Furthermore, since $\left\{-\frac{k}{2\sigma^2}: \sigma^2 > 0\right\}$ contains an open set in $\mathcal{R}$, $T(Y) = \sum_{i=1}^m Y_i$ is a complete sufficient statistic for σ^2 . Therefore, $Y = MSE$ is a complete sufficient statistic for σ^2 (Casella & Berger, 2002).

W_k , as a function of MSE , is the minimum variance unbiased estimator (MVUE) of its expected value Σ_{σ^2} . Therefore, W_k achieves the minimum variance (Casella & Berger, 2002). $\square$

Theorem 5. $\frac{k}{k+2}MSE^2$ converges in probability to σ^4 with any fixed σ^2 , as k goes to infinity under the condition described in **Theorem 1**.

Proof: The expected value of $\frac{k}{k+2}MSE^2$ can be found as follows, where $E(MSE^2)$ is given by proof of **Theorem 3**.

$$\begin{aligned} E\left(\frac{k}{k+2}MSE^2\right) &= \frac{k}{k+2}E(MSE^2) \\ &= \frac{k}{k+2}\left(\frac{2\sigma^4}{k} + \sigma^4\right) \\ &= \frac{2\sigma^4}{k+2} + \frac{k\sigma^4}{k+2} \\ &= \sigma^4 \end{aligned}$$

It is obvious that $\frac{k}{k+2}MSE^2$ is an unbiased estimator for σ^4 . Next, we need to consider its variance.

$$\begin{aligned} \text{Var}\left(\frac{k}{k+2}MSE^2\right) &= \frac{k^2}{(k+2)^2}\text{Var}(MSE^2) \\ &= \frac{k^2}{(k+2)^2}\left(E(MSE^4) - (E(MSE^2))^2\right) \\ &= \frac{k^2}{(k+2)^2}\left(E(MSE^4) - \left(\frac{2\sigma^4}{k} + \sigma^4\right)^2\right) \\ &= \frac{k^2}{(k+2)^2}\left(E(MSE^4) - \frac{(k+2)^2}{k^2}\sigma^8\right) \\ &= \frac{k^2}{(k+2)^2}E(MSE^4) - \sigma^8 \end{aligned}$$

Where

$$\begin{aligned} E(MSE^4) &= E\left(\frac{SSE^4}{k^4}\right) \\ &= E\left(\frac{SSE^4}{\sigma^8} \frac{\sigma^8}{k^4}\right) \\ &= \frac{\sigma^8}{k^4}E\left(\left(\frac{SSE}{\sigma^2}\right)^4\right) \end{aligned}$$

We can find $E\left(\left(\frac{SSE}{\sigma^2}\right)^4\right)$ by using the fact that $\frac{SSE}{\sigma^2} \sim \chi_{(k)}^2$ and the Moment Generating Function of $\chi_{(k)}^2$. We have

$$E\left(\left(\frac{SSE}{\sigma^2}\right)^4\right) = k(k+2)(k+4)(k+6)$$

and

$$E(MSE^4) = \frac{(k+2)(k+4)(k+6)\sigma^8}{k^3}$$

Finally, we have

$$\begin{aligned} \text{Var}\left(\frac{k}{k+2}MSE^2\right) &= \frac{k^2}{(k+2)^2}E(MSE^4) - \sigma^8 \\ &= \frac{(k+4)(k+6)}{k(k+2)}\sigma^8 - \sigma^8 \\ &= \frac{8k+24}{k^2+2k}\sigma^8 \end{aligned}$$

It can be seen that $\text{Var}\left(\frac{k}{k+2}MSE^2\right)$ goes to 0 as k goes to infinity. Therefore, $\frac{k}{k+2}MSE^2$ is a consistent sequence of estimators of σ^4 . Namely, it converges in probability to σ^4 as k goes to infinity (Casella & Berger, 2002). $\square$

Theorem 6. W_k converges in probability to 0 with any fixed σ^2 , as k goes to infinity under the condition described in **Theorem 1**. Namely, W_k and Σ_{σ^2} converge to the same limit.

Proof: It is obvious that $\frac{2}{k}$ converges to 0 everywhere as k goes to infinity (Rudin, 1953), which implies a convergence in probability (Rudin, 1953; Tao, 2011; Casella & Berger, 2002). Furthermore, from **Theorem 5** we know that $\frac{k}{k+2}MSE^2$ converges in probability to σ^4 which implies a convergence in distribution (Casella & Berger, 2002). Therefore, we have the following results by Slutsky's Theorem.

$$W_k = \frac{2}{k} \frac{k}{k+2} MSE^2 \xrightarrow{\text{distribution}} 0 \cdot \sigma^4 = 0$$

A convergence in distribution to a constant implies a convergence in probability to the same constant (Casella & Berger, 2002). Therefore, W_k converges in probability to 0 as k goes to infinity. **Theorem 2** shows Σ_{σ^2} converges to 0 everywhere. Therefore, W_k and Σ_{σ^2} converge to the same limit. $\square$

3. Comparison in MSE_{eval} between S_k and W_k

We name the mean squared error in the sense of statistical inference MSE_{eval} , where “eval” means “evaluator”. It is known that $MSE_{eval}(Es) = \text{Variance}(Es) + (\text{Bias}(Es))^2$ (Casella & Berger, 2002), where Es stands for an estimator. We can find the corresponding MSE_{eval} 's for S_k and W_k , respectively:

$$\begin{aligned}
MSE_{eval}(S_k) &= \text{Variance}(S_k) + (\text{Bias}(S_k))^2 \\
&= \frac{2\sigma^4}{k} + \left(\sigma^2 - \frac{2\sigma^4}{k}\right)^2 \\
&= \frac{2\sigma^4}{k} + \sigma^4 - \frac{4\sigma^6}{k} + \frac{4\sigma^8}{k^2}
\end{aligned}$$

and

$$\begin{aligned}
MSE_{eval}(W_k) &= \text{Variance}(W_k) + (\text{Bias}(W_k))^2 \\
&= \frac{4(k+4)(k+6)\sigma^8}{(k+2)k^3} - \frac{4\sigma^8}{(k+2)^2} + 0 \\
&= \frac{4(k+4)(k+6)\sigma^8}{(k+2)k^3} - \frac{4\sigma^8}{(k+2)^2}
\end{aligned}$$

It can be seen that $MSE_{eval}(S_k)$ converges to σ^4 as k goes to infinity as well as $MSE_{eval}(W_k)$ converges to 0. **Figure A1** clearly demonstrates the pattern of MSE_{eval} for the two estimators. Therefore, W_k is highly preferred for large k . As for small k , W_k is strongly recommended as well because of its unbiasedness. Furthermore, sample size less than 100 is not common in epidemiological studies which yields unreliable results.

Figure A1: Comparison of MSE_{eval} between W_k and S_k

4. Simulation Study

The simulation study was conducted in R and all computation code for reproducing simulation results are provided in section 5 of this document. The values of exposure variable were randomly simulated from $N(0,1)$ distribution. One covariate was generated and the values were from the same distribution. Therefore, p equals 2, and $k = n - 3$ varies according to the number of observations n generated. A $Weibull(0.1, 2)$ distribution was used as the baseline hazards function to generate survival data based on Cox proportional hazard model. Since the Cox model was used, all simulations follow the assumption of rare outcome—death rate is smaller than 15% (Valeri & VanderWeele, 2015; VanderWeele & Ding, 2017). Please note that this rare outcome

assumption can be dropped when using accelerated failure models (Valeri & VanderWeele, 2015). The values of all parameters are listed in **Table A1**.

Table A1: List of parameters

Linear Regression Model			Cox Proportional Hazard Model			
β_0	β_1	β_2	γ_1	γ_2	γ_3	γ_4
0.03	0.04	0.05	$\log(1.1)$	$\log(2)$	$\log(2)$	$\log(1.8)$

Table A2 presents the simulation results. Five different values of σ^2 were chosen from 0.01 through 10, and each value corresponds to a large k value. It can be seen that the standard errors of the exponential part of estimated natural direct effects calculated with W_k are consistently smaller than those with S_k , resulting in much narrower 95% confidence intervals. The 95% confidence intervals with W_k offer much improved precision over that from S_k for varying values of $\sigma^2 = 0.01, 0.1, 1, 5$, and 10. In conclusion, the simulation study clearly states that W_k has more power than S_k to detect mediation effects and surpasses S_k in large measure.

Table A2: Single Simulation Results

	k	σ^2	True natural direct effect	Estimated natural direct effect	Exponential Standard Error	95 C.I. Lower	95 C.I. Upper
W_k	100	0.01	1.1	3.1	0.52	1.1	8.6
S_k					2.4	0.03	358
W_k	150	0.1	1.2	1.8	0.29	1.0	3.1
S_k					0.43	0.76	4.1
W_k	200	1	2.3	2.3	0.27	1.3	3.9
S_k					0.46	0.94	5.6
W_k	300	5	41	37	0.97	5.6	248
S_k					2.1	0.63	2226
W_k	500	10	1515	1548	1.3	125	19108
S_k					2.6	10	239097

The simulation study with 100,000 repetitions was performed for each setting to draw the big picture and to overcome the limitation of random errors due to different seed selection. **Figure A2** presents the results with scatterplots, where the x axis represents the standard error of the exponential part of natural direct effect from W_k as well as the y axis represents that from S_k . It can be seen that all points lie above the $y = x$ line (dashed line), which clearly states that the standard error from our proposed estimator is consistently smaller than that from the previous literature for $k > 100$ in our simulation. **Table A3** presents the numeric results, which supports the same conclusion.

Table A3: Summary of simulation studies with 100,000 repetitions

Settings	Mean of SE from W_k	Mean of SE from S_k	Difference (SE from S_k – SE from W_k)			
			Mean	Median	First Quartile	Third Quartile
$k = 100$ $\sigma^2 = 0.01$	0.41	1.3	0.87	0.22	0.03	0.92
$k = 150$ $\sigma^2 = 0.1$	0.32	0.63	0.31	0.09	0.01	0.38
$k = 200$ $\sigma^2 = 1$	0.43	0.97	0.54	0.48	0.29	0.71
$k = 300$ $\sigma^2 = 5$	1.1	2.1	1.1	1.0	0.86	1.2
$k = 500$ $\sigma^2 = 10$	1.4	2.8	1.4	1.4	1.2	1.6

Figure A2: Scatterplots of Exponential Standard Errors from W_k versus S_k

5. R Code used for simulation

```
#####  
### This R script is for the simulation study of the research  ###  
### letter "An Improvement in Estimation of the Standard Error ###  
### for the Natural Direct Effect in Causal Mediation Analysis"###  
#####  
### R version 3.4.2 (2017-09-28)                                     ###  
#####  
#####  
### The simulation needs the following two packages                ###  
#####  
library(survival) # This is for COX regression                    ###  
library(msm) # This is for Delta Method                          ###  
#####  
  
#####  
### The following values are for all parameters                    ###  
#####  
## The beta values are for the mediator linear regression  
beta_0=0.03  
beta_1=0.04  
beta_2=0.05  
## The gamma values are for the Cox proportional hazard model  
gamma_1=log(1.1)  
gamma_2=log(2)  
gamma_3=log(2)  
gamma_4=log(1.8)  
#####  
  
#####  
### The following functions are for utility purpose                ###  
#####
```

```

## This function generates the true natural direct effect based on
## the parameters, sigma_squared and exposure levels a and
## a_asterisk. The parameters are listed above and do not change.
## The exposure levels a and a_asterisk are set to be 1 and 0,
## respectively.
## Therefore, the only argument is sigma_squared.
## The mean of the covariate is also used but it's zero in this
## situation since the only covariate is sampled from N(0,1)
get.true_nde<-function(sigma_squared){
  return(exp(gamma_1+gamma_3*(beta_0+gamma_2*sigma_squared)+
            0.5*gamma_3^2*sigma_squared))
}

#####
## This function generates the n random samples from a N(0,1)
## distribution, and it will be used for both the exposure a
## and the mediator m.
get.a_and_c<-function(n){
  return(rnorm(n))
}

#####
## This function generates the random samples for the mediator
## based on the given n, sigma_squared, a and c values
## where the mediator is the response variable of a linear model
get.m<-function(n,sigma_squared,a,c){
  return(beta_0+beta_1*a+beta_2*c+rnorm(n,mean=0,
                                         sd=sqrt(sigma_squared)))
}

#####
## This function generates a data frame consists of a,m,a*m and c
## with corresponding names, where a is the exposure, m is the
## mediator, a*m is the interaction term and c is the covariate.
get.data_frame<-function(a,m,c){

```

```

df=data.frame(a,m,a*m,c)

colnames(df)=c("a","m","am","c")

return(df)
}

#####
## This function generates a survival data set based on the data
## frame and a Weibull(0.1,2) distribution.
## The censored column is named status, where 0 indicates censored
## and 1 indicates an event.
simulWeib <- function(N, lambda, rho, x, beta, rateC)
{
  # covariate --> N Bernoulli trials
  # x <- sample(x=c(0, 1), size=N, replace=TRUE, prob=c(0.5, 0.5))

  # Weibull latent event times

  v <- runif(n=N)
  if (length(beta)==1) cmat<-x*beta
  else cmat<- as.matrix(x) %*% matrix(beta,ncol=1)

  Tlat <- (- log(v) / (lambda * exp(cmat)))^(1 / rho)

  # censoring times
  C <- rexp(n=N, rate=rateC)

  # follow-up times and event indicators
  time <- pmin(Tlat, C)
  status <- as.numeric(Tlat <= C)

  # data set
  data.frame(id=1:N,

```

```

        time=time,      # follow up time

        status=status, # vital status

        x=x)

}

#####

## This function gives the ratio of events over total observations
## based on a survival data set generated from the above function.
## Make sure this ratio is below 15% to guarantee the rare-outcome
## assumption.

get.ratio<-function(surv_data){

    return(sum(surv_data$status)/length(surv_data$status))

}

#####

## This function gives the coefficient estimates from the mediator
## linear model and the estimate of the sigma_squared, namely MSE
## and the estimate for Var(MSE) from our proposed estimator.

get.regression<-function(data){

    model=lm(x.m~x.a+x.c,data)

    mse=sum(model$residuals^2)/(length(model$residuals)-3)

    return(list(unnname(model$coefficients[1]),

                unnname(model$coefficients[2]),

                unnname(model$coefficients[3]),

                mse, 2*mse^2/(length(model$residuals)-1),model))

}

#####

## This function gives the coefficient estimates from the COX model

get.cox<-function(data){

    model=coxph(Surv(time, status) ~ x.a*x.m+x.c, data)

    return(list(unnname(model$coefficients[1]),

                unnname(model$coefficients[2]),

                unnname(model$coefficients[4]),model))

}

```

```
#####
## This function gives the estimated natural direct effect
get.estimated_nde<-function(beta_0_h,beta_1_h,beta_2_h,
                             gamma_1_h,gamma_2_h,
                             gamma_3_h,mean_c_h,sigma_squared_h){
  estimated_nde=exp(gamma_1_h+gamma_3_h*(beta_0_h+beta_1_h*0+
                                         beta_2_h*mean_c_h+
                                         gamma_2_h*sigma_squared_h)+
                    0.5*gamma_3_h^2*sigma_squared_h)
}

#####
## This function gives the covariance matrix of those estimators
## by using OUR PROPOSED ESTIMATOR or the estimator from previous
## literature, depending on the third argument of the function.
get.covariance_matrix<-function(linear_model,surv_model,var_mse){
  cov_linear=unname(vcov(linear_model))
  cov_surv=vcov(surv_model)
  cov_surv=matrix(c(cov_surv[1,1],cov_surv[1,2],cov_surv[1,4],
                    cov_surv[2,1],cov_surv[2,2],cov_surv[2,4],
                    cov_surv[4,1],cov_surv[4,2],cov_surv[4,4]),
                  nrow=3,ncol=3,byrow=TRUE)
  part_1=cbind(cov_linear,matrix(0,3,4))
  part_2=cbind(cov_surv,c(0,0,0))
  part_2=cbind(matrix(0,3,3),part_2)
  part_2=rbind(part_2,c(0,0,0,0,0,0,var_mse))
  return(rbind(part_1,part_2))
}

#####
## This function computes the standard error of exponential part of
## natural direct effect by using Delta method
get.exponential_se<-function(mean_h,cov,mean_c_h){
```

```

temp_f=paste("x3*",mean_c_h,sep="")
temp_f=paste("~x4+x6*(x1+x2*0+",temp_f,"+x5*x7)+0.5*(x6)^2*x7",sep="")
f=as.formula(temp_f)
return(deltamethod(f,mean_h,cov))
}

```

```
#####
```

```

## This function is used for a single simulation.
## Three arguments are k, sigma_squared and rate_C.
## rate_C is used to justify the rare outcome assumption
get.single_simulation<-function(k,sigma_squared,rate_C){
  # initialize n
  n=k+3

  # find the true nde
  true_nde=get.true_nde(sigma_squared)
  # generate the random samples for a and c
  a=get.a_and_c(n)
  c=get.a_and_c(n)

  # generate random samples for m
  m=get.m(n,sigma_squared,a,c)

  # generate a data frame to contain these variables generated above
  # and add an interaction term
  df=get.data_frame(a,m,c)

  # generate the survival data set based on the data frame just
  # generated. The baseline function used is Weibull(0.1,2)
  surv_df<-simulWeib(N=n, lambda=0.1,
                     rho=2,
                     beta=c(gamma_1,gamma_2,gamma_3,gamma_4),

```



```

x=df, rateC=rate_C)

# print out the ratio to check
print(get.ratio(surv_df))

# next need to find the estimated nde
# get the needed components from mediation linear regression
m_reg=get.regression(surv_df)
# get the needed components from Cox model
s_reg=get.cox(surv_df)
# get the estimated nde
est_nde=get.estimated_nde(m_reg[[1]],
                          m_reg[[2]],
                          m_reg[[3]],
                          s_reg[[1]],
                          s_reg[[2]],
                          s_reg[[3]],
                          mean(surv_df$x.c),m_reg[[4]])

# Second need to find the exponential standard errors
# with our proposed estimator and previous estimator

# our proposed estimator
our_cov=get.covariance_matrix(m_reg[[6]],s_reg[[4]],m_reg[[5]])
mean_h_our=c(m_reg[[1]],m_reg[[2]],m_reg[[3]],s_reg[[1]],
             s_reg[[2]],s_reg[[3]],m_reg[[4]])
our_se=get.exponential_se(mean_h_our,our_cov,mean(surv_df$x.c))

# previous estimator
pre_cov=get.covariance_matrix(m_reg[[6]],s_reg[[4]],m_reg[[4]])
mean_h_pre=c(m_reg[[1]],m_reg[[2]],m_reg[[3]],s_reg[[1]],
             s_reg[[2]],s_reg[[3]],m_reg[[4]])

```

```

pre_se=get.exponential_se(mean_h_pre,pre_cov,mean(surv_df$x.c))

# find the 95% CI for nde from our proposed estimator
our_expo_l=log(est_nde)-qnorm(0.975)*our_se
our_expo_u=log(est_nde)+qnorm(0.975)*our_se
our_nde_l=exp(our_expo_l)
our_nde_u=exp(our_expo_u)

# find the 95% CI for nde from previous estimator
pre_expo_l=log(est_nde)-qnorm(0.975)*pre_se
pre_expo_u=log(est_nde)+qnorm(0.975)*pre_se
pre_nde_l=exp(pre_expo_l)
pre_nde_u=exp(pre_expo_u)

# Next we can make a data frame for the calculated results
# so that the results can be easily compared
our_result=c(k,sigma_squared,true_nde,est_nde,
             our_se,our_nde_l,our_nde_u)
pre_result=c(k,sigma_squared,true_nde,est_nde,
            pre_se,pre_nde_l,pre_nde_u)

results=rbind(our_result,pre_result)
results=as.data.frame(results)
colnames(results)=c("k","sigma^2","True NDE","Estimated NDE",
                  "Exponential SE","95 CI Lower","95 CI Upper")

return(results)

}

#####

```

```

## This function is for the simulation with 100,000 repetitions
## Please read the comments in this function very carefully!!!
get.massive_simulation<-function(k,sigma_squared,rate_C){
  # set the value of n
  # here n=k+3
  n=k+3

  # compute the true nde here
  true_nde=get.true_nde(sigma_squared)

  # generate the random samples for a and c
  a=get.a_and_c(n)
  c=get.a_and_c(n)

  # generate random samples for m
  m=get.m(n,sigma_squared,a,c)

  # generate a data frame to contain these variables
  # generated above and add an interaction term
  df=get.data_frame(a,m,c)

  ##### IMPORTANT #####
  # Generate the survival data set based on the data frame just #
  # generated. The baseline function used is Weibull(0.1,2). #
  # A while loop is used here to ensure the rare outcome #
  # assumption (ratio<0.15) and to avoid the number of events #
  # that is so small that the survival model cannot be fitted. #
  # Please note that the lower bound of ratio was set to be 0.1 #
  # for the setting of k=100 and sigma_squared=0.01 to adjust #
  # for the small sample size for the 100,000 repetitions. #
  # For all other 100,000-repetition cases, the lower bound of #
  # ratio was set to be 0.08. #

```

```
#####

# set up the initial values to enter the loop
ratio=0.01

# change "ratio<0.08" to "ratio<0.1" for the setting of k=100
# and sigma_squared=0.01.
while(ratio<0.08 | ratio>=0.15){
  # get the simulated survival data
  surv_df<-simulWeib(N=n, lambda=0.1, rho=2,
                    beta=c(gamma_1,gamma_2,gamma_3,gamma_4),
                    x=df, rateC=rate_C)

  # get the ratio to ensure it is in the correct range
  ratio=get.ratio(surv_df)
  # you can print out ratio if you like by uncommenting the next line
  # print(ratio)
}

# next need to find the estimated nde
# get the needed components from mediation linear regression
m_reg=get.regression(surv_df)
# get the needed components from Cox model
s_reg=get.cox(surv_df)
# get the estimated nde
est_nde=get.estimated_nde(m_reg[[1]],
                        m_reg[[2]],
                        m_reg[[3]],
                        s_reg[[1]],
                        s_reg[[2]],
                        s_reg[[3]],mean(surv_df$x.c),m_reg[[4]])
```

```

# Second need to find the exponential standard errors with our
# proposed estimator and previous estimator

# our proposed estimator
our_cov=get.covariance_matrix(m_reg[[6]],s_reg[[4]],m_reg[[5]])
mean_h_our=c(m_reg[[1]],m_reg[[2]],m_reg[[3]],s_reg[[1]],
             s_reg[[2]],s_reg[[3]],m_reg[[4]])
our_se=get.exponential_se(mean_h_our,our_cov,mean(surv_df$x.c))

# previous estimator
pre_cov=get.covariance_matrix(m_reg[[6]],s_reg[[4]],m_reg[[4]])
mean_h_pre=c(m_reg[[1]],m_reg[[2]],m_reg[[3]],s_reg[[1]],
             s_reg[[2]],s_reg[[3]],m_reg[[4]])
pre_se=get.exponential_se(mean_h_pre,pre_cov,mean(surv_df$x.c))

# find the 95% CI for nde from our proposed estimator
our_expo_l=log(est_nde)-qnorm(0.975)*our_se
our_expo_u=log(est_nde)+qnorm(0.975)*our_se
our_nde_l=exp(our_expo_l)
our_nde_u=exp(our_expo_u)

# find the 95% CI for nde from previous estimator
pre_expo_l=log(est_nde)-qnorm(0.975)*pre_se
pre_expo_u=log(est_nde)+qnorm(0.975)*pre_se
pre_nde_l=exp(pre_expo_l)
pre_nde_u=exp(pre_expo_u)

# Next we can make a data frame for the calculated results
# so that the results can be easily compared
our_result=c(k,sigma_squared,true_nde,est_nde,
             our_se,our_nde_l,our_nde_u)

```

```

pre_result=c(k,sigma_squared,true_nde,est_nde,
             pre_se,pre_nde_l,pre_nde_u)

results=rbind(our_result,pre_result)
results=as.data.frame(results)
colnames(results)=c("k","sigma^2","True NDE",
                    "Estimated NDE","Exponential SE",
                    "95 CI Lower","95 CI Upper")

return(results)

}

#####
### Here begins the single simulation process                                     ###
### If you want to retrieve the results in Table 1 and Table A2,###
### please uncomment the seeds we used                                         ###
#####
## The first setting is k=100 and sigma squared = 0.01
## If you want to retrieve the simulation results on the research
## letter, please uncomment set.seed().
#####
#set.seed(-1289684683)
result_k100_ss001=get.single_simulation(100,0.01,1.3)
print(result_k100_ss001)

#####
## The second setting is k=150 and sigma squared = 0.1
#####
#set.seed(-861117810)
result_k150_ss01=get.single_simulation(150,0.1,1.3)

```

```

print(result_k150_ss01)

#####
## The third setting is k=200 and sigma squared = 1
#####
#set.seed(62626208)
result_k200_ss1=get.single_simulation(200,1,1.3)
print(result_k200_ss1)

#####
## The forth setting is k=300 and sigma squared = 5
#####
#set.seed(1794896543)
result_k300_ss5=get.single_simulation(300,5,1.3)
print(result_k300_ss5)

#####
## The fifth setting is k=500 and sigma squared = 10
#####
#set.seed(-1570072643)
# 1.7 is used for rateC for the fifth setting
# to justify for the rare outcome assumption
result_k500_ss10=get.single_simulation(500,10,1.7)
print(result_k500_ss10)

#####
### Here begins the 100,000-repetition simulation          ###
### If you want to retrieve the results in Table A3,      ###
### please uncomment the seeds we used.                  ###
#####
## The first setting is k=100 and sigma squared = 0.01
#####

```

```

# initialize three vectors to store the se's and estimated nde
est_nde_1=c(NULL)
our_se_vec_1=c(NULL)
pre_se_vec_1=c(NULL)

# randomly generate a seed
seed=runif(1,-1,1)*1000000000
print(seed)
set.seed(seed)

# if you want to retrieve the simulation results on paper,
# please uncomment the next line
# set.seed(373907661)

# start the for loop to run 100,000 repetitions
for(i in 1:100000){
  # use the same function from utility
  result_k100_ss001=get.massive_simulation(100,0.01,1.3)
  # save the estimated nde
  est_nde_1[i]=result_k100_ss001[1,4]
  # save the se from our estimator
  our_se_vec_1[i]=result_k100_ss001[1,5]
  # save the se from previous estimator
  pre_se_vec_1[i]=result_k100_ss001[2,5]
  print(i)
}

# calculate the difference between previous se and our se
difference_1=pre_se_vec_1-our_se_vec_1

#####

```



```

## The second setting is k=150 and sigma squared = 0.1
#####

# initialize three vectors to store the se's and estimated nde
est_nde_2=c(NULL)
our_se_vec_2=c(NULL)
pre_se_vec_2=c(NULL)

# randomly generate a seed
seed=runif(1,-1,1)*1000000000
print(seed)
set.seed(seed)

# if you want to retrieve the simulation results on paper,
# please uncomment the next line
# set.seed(-27221047)

# start the for loop to run 100,000 repetitions
for(i in 1:100000){
  # use the same function from utility
  result_k150_ss01=get.massive_simulation(150,0.1,1.3)
  # save the estimated nde
  est_nde_2[i]=result_k150_ss01[1,4]
  # save the se from our estimator
  our_se_vec_2[i]=result_k150_ss01[1,5]
  # save the se from previous estimator
  pre_se_vec_2[i]=result_k150_ss01[2,5]
  print(i)
}

# calculate the difference between previous se and our se
difference_2=pre_se_vec_2-our_se_vec_2

```

```
#####
## The third setting is k=200 and sigma squared = 1
#####
# initialize three vectors to store the se's and estimated nde
est_nde_3=c(NULL)
our_se_vec_3=c(NULL)
pre_se_vec_3=c(NULL)

# randomly generate a seed
seed=runif(1,-1,1)*1000000000
print(seed)
set.seed(seed)

# if you want to retrieve the simulation results on paper,
# please uncomment the next line
#set.seed(88520570)

# start the for loop to run 100,000 repetitions
for(i in 1:100000){
  # use the same function from utility
  result_k200_ss1=get.massive_simulation(200,1,1.3)
  # save the estimated nde
  est_nde_3[i]=result_k200_ss1[1,4]
  # save the se from our estimator
  our_se_vec_3[i]=result_k200_ss1[1,5]
  # save the se from previous estimator
  pre_se_vec_3[i]=result_k200_ss1[2,5]
  print(i)
}

# calculate the difference between previous se and our se
difference_3=pre_se_vec_3-our_se_vec_3
```

```
#####
## The fourth setting is k=300 and sigma squared = 5
#####
# initialize three vectors to store the se's and estimated nde
est_nde_4=c(NULL)
our_se_vec_4=c(NULL)
pre_se_vec_4=c(NULL)

# randomly generate a seed
seed=runif(1,-1,1)*1000000000
print(seed)
set.seed(seed)

# if you want to retrieve the simulation results on paper,
# please uncomment the next line
# set.seed(-433938124)

# start the for loop to run 100,000 repetitions
for(i in 1:100000){
  # use the same function from utility
  result_k300_ss5=get.massive_simulation(300,5,1.3)
  # save the estimated nde
  est_nde_4[i]=result_k300_ss5[1,4]
  # save the se from our estimator
  our_se_vec_4[i]=result_k300_ss5[1,5]
  # save the se from previous estimator
  pre_se_vec_4[i]=result_k300_ss5[2,5]
  print(i)
}

# calculate the difference between previous se and our se
```

```

difference_4=pre_se_vec_4-our_se_vec_4

#####
## The fifth setting is k=500 and sigma squared = 10
#####
# initialize three vectors to store the se's and estimated nde
est_nde_5=c(NULL)
our_se_vec_5=c(NULL)
pre_se_vec_5=c(NULL)

# randomly generate a seed
seed=runif(1,-1,1)*1000000000
print(seed)
set.seed(seed)

# if you want to retrieve the simulation results on paper,
# please uncomment the next line
# set.seed(639022465)

# start the for loop to run 100,000 repetitions
for(i in 1:100000){
  # use the same function from utility
  # The rate_C is set to be 1.7 here to justify for the rare
  # outcome assumption
  result_k500_ss10=get.massive_simulation(500,10,1.7)
  # save the estimated nde
  est_nde_5[i]=result_k500_ss10[1,4]
  # save the se from our estimator
  our_se_vec_5[i]=result_k500_ss10[1,5]
  # save the se from previous estimator
  pre_se_vec_5[i]=result_k500_ss10[2,5]
  print(i)
}

```

```
}
```

```
# calculate the difference between previous se and our se  
difference_5=pre_se_vec_5-our_se_vec_5
```

References

- Casella, G., & Berger, R. L. (2002). *Statistical Inference*. Pacific Grove: Duxbury.
- Christensen, R. (2011). *Plane Answers to Complex Questions: the Thoery of Linear Models*. New York: Springer New York.
- Rudin, W. (1953). *Principles of Mathematical Analysis*. New York: McGraw-Hill Book.
- Tao, T. (2011). *An Introduction to Measure Theory*. Providence, R.I.: American Mathematical Society.
- Valeri, L., & VanderWeele, T. J. (2013). Mediation analysis allowing for exposure-mediator interactions and causal interpretation: theoretical assumptions and implementation with SAS and SPSS macros. *Psychol Methods*, 18, 137-150.
- Valeri, L., & VanderWeele, T. J. (2015). SAS macro for causal mediation analysis with survival data. *Epidemiology*, 26, e23-e24.
- VanderWeele, T. J., & Ding, P. (2017). Sensitivity Analysis in Observational Research: Introducing the E-Value. *Annals of Internal Medicine*, 167, 268-274.