Appendix: Implementation Code

```
## install packages
# install.packages("augSIMEX")
# install.packages("scales")
# install.packages("microbenchmark")
# install.packages("ggplot2")
library(augSIMEX)
library(scales)
library(microbenchmark)
library(ggplot2)



## Seed Generator
set.seed(2018)
Seeds <- sample(100000,size=1000,replace=FALSE)
options(warn=-1)

source("DataGenerator.R")

######################
## Section 5.1 Bias ##
## This Section will run approximately [6 hours] without parellal computing

Results_Table1 <- function(i,rate){
  tryCatch({
    Data <- DataGenerator(Seeds[i],rate,"1D")
    start <- proc.time()

    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar+W+Zstar,
          mismodel=pi|qi~1, family = binomial,
          data=Data$Main, validationdata=Data$Validation, subset=NULL,
          err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
          err.mat = NULL,
          lambda=NULL, M=5, B=20, nBoot=2, extrapolation=c("both"))
    ## Results without accounting for measurement error and misclassification
    naive <- glm(Y~Xstar+W+Zstar,family = binomial,data=Data$Main)
    end <- proc.time()-start
```

```
    coefficients <- Re$coefficients
    se <- (Re$vcov)
    return(list(coefs.naive=naive$coefficients,
          coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
          coefs.quadra=coefficients[1:(length(coefficients)/2)],
          se.naive=sqrt(diag(vcov(naive))),
          se.linear=se[(length(coefficients)/2+1):length(coefficients)],
          se.quadra=se[1:(length(coefficients)/2)],
          time=end[1]))
  }, error = function(e) return(NA))
}

Table1_1 <- lapply(1:1000,FUN=Results_Table1,rate=0.1)
Table1_3 <- lapply(1:1000,FUN=Results_Table1,rate=0.3)
Table1_5 <- lapply(1:1000,FUN=Results_Table1,rate=0.5)

save(Table1_1,file="Table1_1.RData")
save(Table1_3,file="Table1_3.RData")
save(Table1_5,file="Table1_5.RData")



######################
## Evaluation and rearrange to obtain Table 1

## Function of computing performance results from linear and quadratic extrepolation
get_naive_Table1 <- function(data,truebeta){
  pooled_form <- unlist(lapply(data,FUN = function(x){
    if (is.na(x)) return(NA) else return(x$coefs.naive)}))
  matrix_form <- matrix(pooled_form,nrow=1000,byrow = T)
  colnames(matrix_form) <- c("(Intercept)","X","W","Z")
  matrix_form <- matrix_form[,c("(Intercept)","X","Z","W")]
  mean <- colMeans(matrix_form,na.rm = T)
  bias <- abs(mean-truebeta)
  emperical_sd <- apply(matrix_form,MARGIN=2,FUN=sd, na.rm = T)
  output <- cbind(bias,emperical_sd)
  return(as.matrix(output))
}
```

```
get_linear_Table1 <- function(data,truebeta){
 pooled_form <- unlist(lapply(data,FUN = function(x){
   if (is.na(x)) return(NA) else return(x$coefs.linear)}))
 matrix_form <- matrix(pooled_form,nrow=1000,byrow = T)
 colnames(matrix_form) <- names(data[[1]]$coefs.linear)
 matrix_form <- matrix_form[,c("(Intercept)","X","Z","W")]
 mean <- colMeans(matrix_form,na.rm = T)
 bias <- abs(mean-truebeta)
 emperical_sd <- apply(matrix_form,MARGIN=2,FUN=sd, na.rm = T)
 output <- cbind(bias,emperical_sd)
 return(as.matrix(output))
}

get_quadra_Table1 <- function(data,truebeta){
  pooled_form <- unlist(lapply(data,FUN = function(x){
    if (is.na(x)) return(NA) else return(x$coefs.quadra)}))
  matrix_form <- matrix(pooled_form,nrow=1000,byrow = T)
  colnames(matrix_form) <- names(data[[1]]$coefs.quadra)
  matrix_form <- matrix_form[,c("(Intercept)","X","Z","W")]
  mean <- colMeans(matrix_form,na.rm = T)
  bias <- abs(mean-truebeta)
  emperical_sd <- apply(matrix_form,MARGIN=2,FUN=sd ,na.rm = T)
  output <- cbind(bias,emperical_sd)
  return(as.matrix(output))
}

True_beta_Table1 <- c(0.1,-1,0.7,0.5)

Table1_1_naive <- get_naive_Table1(Table1_1,True_beta_Table1)
Table1_1_linear <- get_linear_Table1(Table1_1,True_beta_Table1)
Table1_1_quadratic <- get_quadra_Table1(Table1_1,True_beta_Table1)
Table1_3_naive <- get_naive_Table1(Table1_3,True_beta_Table1)
Table1_3_linear <- get_linear_Table1(Table1_3,True_beta_Table1)
Table1_3_quadratic <- get_quadra_Table1(Table1_3,True_beta_Table1)
Table1_5_naive <- get_naive_Table1(Table1_5,True_beta_Table1)
Table1_5_linear <- get_linear_Table1(Table1_5,True_beta_Table1)
Table1_5_quadratic <- get_quadra_Table1(Table1_5,True_beta_Table1)
```

```
Table1 <- cbind(rbind(Table1_1_naive,Table1_3_naive,Table1_5_naive),
              rbind(Table1_1_linear,Table1_3_linear,Table1_5_linear),
              rbind(Table1_1_quadratic,Table1_3_quadratic,Table1_5_quadratic))
Table1 <- round(Table1,3)
Table1

######################
## Section 5.2 variance estimation ##
## This Section will run approximately [6 hours] without parellal computing

Results_Table2_1D <- function(i){
  tryCatch({
    start <- proc.time()
    Data <- DataGenerator(Seeds[i],0.5,"1D")

    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar+W+Zstar,
              mismodel=pi|qi~1, family = binomial,
              data=Data$Main,validationdata=Data$Validation, subset=NULL,
              err.var="Xstar", mis.var="Zstar", err.true="X",
              mis.true="Z", err.mat = NULL,
              lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"))
    ## Results without accounting for measurement error and misclassification
    naive <- glm(Y~Xstar+W+Zstar,family = binomial,data=Data$Main)
    coefficients <- Re$coefficients
    se <- (Re$vcov)
  }, error = function(e) return(NA))
  end <- proc.time()-start

  return(list(coefs.naive=naive$coefficients,
         coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
         coefs.quadra=coefficients[1:(length(coefficients)/2)],
         se.naive=sqrt(diag(vcov(naive))),
         se.linear=se[(length(coefficients)/2+1):length(coefficients)],
         se.quadra=se[1:(length(coefficients)/2)],
         time=end[1]))
}
```

```
Results_Table2_2D <- function(i){
  tryCatch({
    start <- proc.time()
    Data <- DataGenerator(Seeds[i],0.5,"2D")
    meformula <- Xstar.X1|Xstar.X2~-1+X.X1|-1+X.X2

    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
         pimodel=pi|qi~1, family = binomial,
        data=Data$Main,meformula=meformula,
        validationdata=Data$Validation, subset=NULL,
        err.var=c("Xstar.X1","Xstar.X2"), mis.var="Zstar",
        err.true=c("X.X1","X.X2"), mis.true="Z", err.mat = NULL,
        lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"))
    ## Results without accounting for measurement error and misclassification
    naive <- glm(Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
         family = binomial,data=Data$Main)
    coefficients <- Re$coefficients
    se <- (Re$vcov)
  }, error = function(e) return(NA))
  end <- proc.time()-start

  return(list(coefs.naive=naive$coefficients,
        coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
        coefs.quadra=coefficients[1:(length(coefficients)/2)],
        se.naive=sqrt(diag(vcov(naive))),
        se.linear=se[(length(coefficients)/2+1):length(coefficients)],
        se.quadra=se[1:(length(coefficients)/2)],
        time=end[1]))
}


Results_Table2_RM <- function(i){
  tryCatch({
    start <- proc.time()
    Data <- DataGenerator(Seeds[i],0.5,"RM")

    ## The main function of correcting measurement error and misclassification
```

```
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=logit),
           data=Data$Main,validationdata=Data$Validation, subset=NULL,
           err.var="Xstar1", mis.var="Zstar", err.true="X",
           mis.true="Z", err.mat = NULL,
           repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
           lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"))

    ## Results without accounting for measurement error and misclassification
    naive <- glm(Y~Xstar1+Zstar+W,family = binomial,data=Data$Main)
    coefficients <- Re$coefficients
    se <- (Re$vcov)
  }, error = function(e) return(NA))
  end <- proc.time()-start

  return(list(coefs.naive=naive$coefficients,
        coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
        coefs.quadra=coefficients[1:(length(coefficients)/2)],
        se.naive=sqrt(diag(vcov(naive))),
        se.linear=se[(length(coefficients)/2+1):length(coefficients)],
        se.quadra=se[1:(length(coefficients)/2)],
        time=end[1]))
}

Table2_1D <- lapply(1:1000,FUN=Results_Table2_1D)
Table2_2D <- lapply(1:1000,FUN=Results_Table2_2D)
Table2_RM <- lapply(1:1000,FUN=Results_Table2_RM)



######################
## Evaluation and rearrangement of obtaining Table 2
## Function of computing performance results from linear
##    and quadratic extrepolation
data.var.order <- c("(Intercept)","X","W","Z")
calibration.var.order <- c("(Intercept)","X","Z","W")
coef.object <- "coefs.naive"
se.object <- "se.naive"

get_Table2_unit <- function(data,truebeta,coef.object,se.object,
```

```
                            data.var.order,calibration.var.order){
  #### Rearrange the data of naive study
  ## Bias
  beta_pooled <- unlist(lapply(data,FUN = function(x){
    if (is.na(x)) return(NA) else return(x[[coef.object]])}))
  beta_matrix <- matrix(beta_pooled,nrow=1000,byrow = T)
  colnames(beta_matrix) <- data.var.order
  beta_matrix <- beta_matrix[,calibration.var.order]
  mean <- colMeans(beta_matrix,na.rm = T)
  bias <- abs(mean-truebeta)

  ## Standard Error
  se_pooled <- unlist(lapply(data,FUN = function(x){
    if (is.na(x)) return(NA) else return(x[[se.object]])}))
  se_matrix <- matrix(se_pooled,nrow=1000,byrow = T)
  colnames(se_matrix) <- data.var.order
  se_matrix <- se_matrix[,calibration.var.order]
  se <- colMeans(se_matrix,na.rm = T)

  ## Coverage Rate
  coverage <- lapply(1:1000,FUN = function(i){
    if (is.na(data[[i]])) return(NA) else{
      LB <- data[[i]][[coef.object]]-1.96*data[[i]][[se.object]]
      UB <- data[[i]][[coef.object]]+1.96*data[[i]][[se.object]]
      names(LB) <- names(UB) <- data.var.order
      LB <- LB[calibration.var.order]  # reorder upper bound
      UB <- UB[calibration.var.order]  # reorder lower bound
      return(truebeta>=LB & truebeta<=UB)
    }
  })
  CR <- colMeans(matrix(unlist(coverage),nrow=1000,byrow=T),na.rm=T)


  output <- data.frame(bias=round(bias,3),se=round(se,3),CR=percent(CR))
  return(as.matrix(output))
}

True_beta_1D <- c(0.1,-1,0.7,0.5)
Table2_1D_naive <- get_Table2_unit(Table2_1D, True_beta_1D,
```

```
          "coefs.naive","se.naive",
          data.var.order = c("(Intercept)","X","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))
Table2_1D_linear <- get_Table2_unit(Table2_1D, True_beta_1D,
         "coefs.linear","se.linear",
          data.var.order = c("X","(Intercept)","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))
Table2_1D_quadra <- get_Table2_unit(Table2_1D, True_beta_1D,
          "coefs.quadra","se.quadra",
          data.var.order = c("X","(Intercept)","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))

True_beta_2D <- c(0.1,-1,0.6,0.7,-0.5,1)
Table2_2D_naive <- get_Table2_unit(Table2_2D, True_beta_2D,
          "coefs.naive","se.naive",
          data.var.order = c("(Intercept)","X1","X2","Z","W1","W2"),
          calibration.var.order = c("(Intercept)","X1","X2","Z","W1","W2"))
Table2_2D_linear <- get_Table2_unit(Table2_2D, True_beta_2D,
          "coefs.linear","se.linear",
          data.var.order = c("X1","X2","(Intercept)","W1","W2","Z"),
          calibration.var.order = c("(Intercept)","X1","X2","Z","W1","W2"))
Table2_2D_quadra <- get_Table2_unit(Table2_2D, True_beta_2D,
          "coefs.quadra","se.quadra",
          data.var.order = c("X1","X2","(Intercept)","W1","W2","Z"),
          calibration.var.order =  c("(Intercept)","X1","X2","Z","W1","W2"))

True_beta_1D <- c(0.1,-1,0.7,0.5)
Table2_RM_naive <- get_Table2_unit(Table2_RM, True_beta_1D,
         "coefs.naive","se.naive",
          data.var.order = c("(Intercept)","X","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))
Table2_RM_linear <- get_Table2_unit(Table2_RM, True_beta_1D,
         "coefs.linear","se.linear",
          data.var.order = c("X","(Intercept)","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))
Table2_RM_quadra <- get_Table2_unit(Table2_RM, True_beta_1D,
         "coefs.quadra","se.quadra",
          data.var.order = c("X","(Intercept)","W","Z"),
          calibration.var.order = c("(Intercept)","X","Z","W"))
```

```
Table2 <- cbind(rbind(Table2_1D_naive,Table2_2D_naive,Table2_RM_naive),
              rbind(Table2_1D_linear,Table2_2D_linear,Table2_RM_linear),
              rbind(Table2_1D_quadra,Table2_2D_quadra,Table2_RM_quadra))
Table2


######################
## Section 5.3 variance estimation ##
## This Section will run approximately [6 hours] without parellal computing
Results_Table3_probit_parallel<-function(i){
  tryCatch({
    library(augSIMEX)
    source("DataGenerator.R")

    set.seed(2018)
    Seeds<-sample(100000,size=1000,replace=FALSE)

    start<-proc.time()
    Data <- DataGenerator2(Seeds[i],0.2,"1D",nm=1000,nv=500,link="probit")

    # truemodel <- glm(Y~X1+Z+W1,family = binomial(link=probit))
    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1,
                   family = binomial(link=probit),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"),
                   solvefun = "nleqslv")
    ## Results without accounting for measurement error and misclassification
    naive<-glm(Y~Xstar+W+Zstar,family = binomial(link=probit),data=Data$Main)
    end<-proc.time()-start

    coefficients <- Re$coefficients
    se <- Re$vcov
    return(list(coefs.naive=naive$coefficients,
```

```
                coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
                coefs.quadra=coefficients[1:(length(coefficients)/2)],
                se.naive=sqrt(diag(vcov(naive))),
                se.linear=se[(length(coefficients)/2+1):length(coefficients)],
                se.quadra=se[1:(length(coefficients)/2)],
                time=end[1]))
  }, error = function(e) return(NA))
}



Results_Table3_poisson_parallel<-function(i){
  tryCatch({
    library(augSIMEX)
    source("DataGenerator.R")

    set.seed(2018)
    Seeds<-sample(100000,size=1000,replace=FALSE)

    start<-proc.time()
    Data <- DataGenerator2(Seeds[i],0.2,"1D",nm=1000,nv=500,link="log")
    true <- glm(Y~X1+W1+Z,family = poisson(link=log),data=Data$TrueData)

    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1,
                   family = poisson(link=log),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   lambda=NULL, M=15, B=20, nBoot=50, extrapolation=c("both"),
                   solvefun = "nleqslv")
    ## Results without accounting for measurement error and misclassification
    naive<-glm(Y~Xstar+W+Zstar,family = poisson(link=log),data=Data$Main)
    end<-proc.time()-start

    coefficients <- Re$coefficients
    se <- Re$vcov
    return(list(coefs.naive=naive$coefficients,
                coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
```

```
                coefs.quadra=coefficients[1:(length(coefficients)/2)],
                se.naive=sqrt(diag(vcov(naive))),
                se.linear=se[(length(coefficients)/2+1):length(coefficients)],
                se.quadra=se[1:(length(coefficients)/2)],
                time=end[1]))
  }, error = function(e) return(NA))
}


Results_Table3_probit_repeat<-function(i){
  tryCatch({
    library(augSIMEX)
    source("DataGenerator.R")

    set.seed(2018)
    Seeds<-sample(100000,size=1000,replace=FALSE)

    start<-proc.time()
    Data <- DataGenerator2(Seeds[i],0.2,"RM",nm=1000,nv=500,link="probit")

    # truemodel <- glm(Y~X1+Z+W1,family = binomial(link=probit))
    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=probit),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
                   lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"),
                   solvefun ="nleqslv")
    ## Results without accounting for measurement error and misclassification
    naive<-glm(Y~Xstar1+W+Zstar,family = binomial(link=probit),data=Data$Main)
    end<-proc.time()-start

    coefficients <- Re$coefficients
    se <- Re$vcov
    return(list(coefs.naive=naive$coefficients,
                coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
                coefs.quadra=coefficients[1:(length(coefficients)/2)],
                se.naive=sqrt(diag(vcov(naive))),
```

```
                se.linear=se[(length(coefficients)/2+1):length(coefficients)],
                se.quadra=se[1:(length(coefficients)/2)],
                time=end[1]))
  }, error = function(e) return(NA))
}

Results_Table3_poisson_repeat<-function(i){
  tryCatch({
    library(augSIMEX)
    source("DataGenerator.R")

    set.seed(2018)
    Seeds<-sample(100000,size=1000,replace=FALSE)

    start<-proc.time()
    Data <- DataGenerator2(Seeds[i],0.2,"RM",nm=1000,nv=500,link="log")
    true <- glm(Y~X1+W1+Z,family = poisson(link=log),data=Data$TrueData)

    ## The main function of correcting measurement error and misclassification
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = poisson(link=log),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
                   lambda=NULL, M=5, B=20, nBoot=50, extrapolation=c("both"),
                   solvefun ="nleqslv")
    ## Results without accounting for measurement error and misclassification
    naive<-glm(Y~Xstar1+W+Zstar,family = poisson(link=log),data=Data$Main)
    end<-proc.time()-start

    coefficients <- Re$coefficients
    se <- Re$vcov
    return(list(coefs.naive=naive$coefficients,
                coefs.linear=coefficients[(length(coefficients)/2+1):length(coefficients)],
                coefs.quadra=coefficients[1:(length(coefficients)/2)],
                se.naive=sqrt(diag(vcov(naive))),
                se.linear=se[(length(coefficients)/2+1):length(coefficients)],
                se.quadra=se[1:(length(coefficients)/2)],
                time=end[1]))
```

```
  }, error = function(e) return(NA))
}


Table3_pr <- parLapply(cl, 1:1000, Results_Table3_probit_parallel)
save(Table3_pr,file="Table3_pr_nle2.RData")
Table3_po <- parLapply(cl, 1:1000, Results_Table3_poisson_parallel)
save(Table3_po,file="Table3_po_nle2.RData")
Table3rep_pr <- parLapply(cl, 1:1000, Results_Table3_probit_repeat)
save(Table3rep_pr,file="Table3_pr_rep.RData")
Table3rep_po <- parLapply(cl, 1:1000, Results_Table3_poisson_repeat)
save(Table3rep_po,file="Table3_po_rep.RData")




######################
## Evaluation and rearrangement of obtaining Table 3
## Function of computing performance results from linear and quadratic extrepolation
data.var.order <- c("(Intercept)","X","W","Z")
calibration.var.order <- c("(Intercept)","X","Z","W")
coef.object <- "coefs.naive"
se.object <- "se.naive"

get_Table3_unit <- function(data,truebeta,coef.object,se.object,
                            data.var.order,calibration.var.order){
  #### Rearrange the data of naive study
  ## Bias
  beta_pooled <- unlist(lapply(data,FUN = function(x){
    if (is.na(x)) return(rep(NA,length(data.var.order))) else return(x[[coef.object]])}))
  beta_matrix <- matrix(beta_pooled,nrow=1000,byrow = T)
  colnames(beta_matrix) <- data.var.order
  beta_matrix <- beta_matrix[,calibration.var.order]
  mean <- colMeans(beta_matrix,na.rm = T)
  bias <- abs(mean-truebeta)

  ## Standard Error
  se_pooled <- unlist(lapply(data,FUN = function(x){
```

```
    if (is.na(x)) return(rep(NA,length(data.var.order))) else return(x[[se.object]])}))
  se_matrix <- matrix(se_pooled,nrow=1000,byrow = T)
  colnames(se_matrix) <- data.var.order
  se_matrix <- se_matrix[,calibration.var.order]
  se <- colMeans(se_matrix,na.rm = T)

  ## Coverage Rate
  coverage <- lapply(1:1000,FUN = function(i){
    if (is.na(data[[i]])) return(rep(NA,length(data.var.order))) else{
      LB <- data[[i]][[coef.object]]-1.96*data[[i]][[se.object]]
      UB <- data[[i]][[coef.object]]+1.96*data[[i]][[se.object]]
      names(LB) <- names(UB) <- data.var.order
      LB <- LB[calibration.var.order]  # reorder upper bound
      UB <- UB[calibration.var.order]  # reorder lower bound
      return(truebeta>=LB & truebeta<=UB)
    }
  })
  CR <- colMeans(matrix(unlist(coverage),nrow=1000,byrow=T),na.rm=T)


  output <- data.frame(bias=round(bias,3),se=round(se,3),CR=percent(CR))
  return(as.matrix(output))
}

True_beta <- c(0.1,-1,0.7,0.5)
Table3_ga_naive <- get_Table3_unit(Table3_ga, True_beta,"coefs.naive","se.naive",
                                   data.var.order = c("(Intercept)","X","W","Z"),
                                   calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_ga_linear <- get_Table3_unit(Table3_ga, True_beta, "coefs.linear","se.linear",
                                    data.var.order = c("X","(Intercept)","W","Z"),
                                    calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_ga_quadra <- get_Table3_unit(Table3_ga, True_beta, "coefs.quadra","se.quadra",
                                    data.var.order = c("X","(Intercept)","W","Z"),
                                    calibration.var.order = c("(Intercept)","X","Z","W"))

Table3_pr_naive <- get_Table3_unit(Table3_pr, True_beta,"coefs.naive","se.naive",
                                   data.var.order = c("(Intercept)","X","W","Z"),
                                   calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_pr_linear <- get_Table3_unit(Table3_pr, True_beta, "coefs.linear","se.linear",
```

```
                                    data.var.order = c("X","(Intercept)","W","Z"),
                                    calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_pr_quadra <- get_Table3_unit(Table3_pr, True_beta, "coefs.quadra","se.quadra",
                                    data.var.order = c("X","(Intercept)","W","Z"),
                                    calibration.var.order =  c("(Intercept)","X","Z","W"))

Table3rep_pr_naive <- get_Table3_unit(Table3rep_pr, True_beta,"coefs.naive","se.naive",
                                data.var.order = c("(Intercept)","Xstar1","W","Zstar"),
                                calibration.var.order = c("(Intercept)","Xstar1","Zstar","W"))
Table3rep_pr_linear <- get_Table3_unit(Table3rep_pr, True_beta, "coefs.linear","se.linear",
                                data.var.order = c("X","(Intercept)","W","Z"),
                                calibration.var.order = c("(Intercept)","X","Z","W"))
Table3rep_pr_quadra <- get_Table3_unit(Table3rep_pr, True_beta, "coefs.quadra","se.quadra",
                                data.var.order = c("X","(Intercept)","W","Z"),
                                calibration.var.order =  c("(Intercept)","X","Z","W"))

Table3_po_naive <- get_Table3_unit(Table3_po, True_beta,"coefs.naive","se.naive",
                                data.var.order = c("(Intercept)","X","W","Z"),
                                calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_po_linear <- get_Table3_unit(Table3_po, True_beta, "coefs.linear","se.linear",
                                data.var.order = c("X","(Intercept)","W","Z"),
                                calibration.var.order = c("(Intercept)","X","Z","W"))
Table3_po_quadra <- get_Table3_unit(Table3_po, True_beta, "coefs.quadra","se.quadra",
                                data.var.order = c("X","(Intercept)","W","Z"),
                                calibration.var.order = c("(Intercept)","X","Z","W"))

Table3rep_po_naive <- get_Table3_unit(Table3rep_po, True_beta,"coefs.naive","se.naive",
                               data.var.order = c("(Intercept)","Xstar1","W","Zstar"),
                               calibration.var.order = c("(Intercept)","Xstar1","Zstar","W"))
Table3rep_po_linear <- get_Table3_unit(Table3rep_po, True_beta, "coefs.linear","se.linear",
                               data.var.order = c("X","(Intercept)","W","Z"),
                               calibration.var.order = c("(Intercept)","X","Z","W"))
Table3rep_po_quadra <- get_Table3_unit(Table3rep_po, True_beta, "coefs.quadra","se.quadra",
                               data.var.order = c("X","(Intercept)","W","Z"),
                               calibration.var.order =  c("(Intercept)","X","Z","W"))


Table3_po_quadra_new <- ifelse(abs(Table3_po_quadra)<8,Table3_po_quadra,NA)
mean <- colMeans(Table3_po_quadra_new,na.rm = T)
```

```
Table3 <- cbind(rbind(Table3_pr_naive,Table3rep_pr_naive,Table3_po_naive,Table3rep_po_naive),
 rbind(Table3_pr_linear,Table3rep_pr_linear,Table3_po_linear,Table3rep_po_linear),
 rbind(Table3_pr_quadra,Table3rep_pr_quadra,Table3_po_quadra,Table3rep_po_quadra))




######################
## Section 6 Example ##

data(GeneUni)
example1_naive <- lm(rs38916331 ~ Tibia_M + Batch_O + Weight,  data = GeneUni$Main)
summary(example1_naive)

# Call:
#   lm(formula = rs38916331 ~ Tibia_M + Batch_O + Weight, data = GeneUni$Main)
#
# Residuals:
#   Min      1Q  Median      3Q     Max
# -0.8754 -0.8588  0.1337  0.1423  1.1460
#
# Coefficients:
#   Estimate Std. Error t value Pr(>|t|)
# (Intercept)  0.8994158  0.8581227   1.048    0.295
# Tibia_M      0.0002329  0.0489492   0.005    0.996
# Batch_O      0.0009394  0.0681910   0.014    0.989
# Weight      -0.0015747  0.0093007  -0.169    0.866
#
# Residual standard error: 0.6859 on 750 degrees of freedom
# Multiple R-squared:  4.314e-05,        Adjusted R-squared:  -0.003957
# F-statistic: 0.01079 on 3 and 750 DF,  p-value: 0.9985

set.seed(2018)
example1<-augSIMEX(mainformula = rs38916331 ~ Tibia_M + Batch_O + Weight,
           family = gaussian(link = "identity"),
                   mismodel = pi|qi ~ 1, meformula = Tibia_M ~ Tibia_T,
                   data = GeneUni$Main,validationdata = GeneUni$Validation, subset = NULL,
                   err.var = "Tibia_M", mis.var = "Batch_O", err.true = "Tibia_T",
```

```
                  mis.true = "Batch_T", err.mat = NULL, solvefun = "nleqslv",
                  lambda = NULL, M = 10, B = 20, nBoot = 50, extrapolation="quadratic",
                  initial = c(0,0,0,0,0.01))
summary(example1)

# Call:
#   augSIMEX(mainformula = rs38916331 ~ Tibia_M + Batch_O + Weight,
#            mismodel = pi | qi ~ 1, meformula = Tibia_M ~ Tibia_T,
#            family = gaussian(link = "identity"),
#            data = GeneUni$Main, validationdata = GeneUni$Validation,
#            err.var = "Tibia_M", mis.var = "Batch_O", err.true = "Tibia_T",
#            mis.true = "Batch_T", err.mat = NULL, subset = NULL, lambda = NULL,
#            M = 10, B = 20, nBoot = 50, extrapolation = "quadratic",
#            solvefun = "nleqslv", initial = c(0, 0, 0, 0, 0.01))
#
#
# The error-prone variable: Tibia_M
# The corresponding true variable: Tibia_T
# Number of simulations: 20
# Number of iterations in bootstrap: 50
#
#
# Coefficients:
#   Estimate Std. Error t value Pr(>|t|)
# (Intercept)  0.988156   1.041682   0.949    0.343
# Tibia_T     -0.003697   0.061068  -0.061    0.952
# Weight      -0.001831   0.012590  -0.145    0.884
# Batch_T      0.002227   0.143170   0.016    0.988
#
#
# (Dispersion parameter for gaussian family taken to be NULL)
#
# Null deviance: 352.89  on 753  degrees of freedom
# Residual deviance: 352.96  on 750  degrees of freedom
# AIC: 1577.4


plot(example1,ylim = c(-0.01,0.01) )
```

```
#### Example 2: Repeated Measurements
set.seed(2018)
data(GeneRepeat)
example2<-augSIMEX(mainformula = rs223979909 ~ Totdist + Batch_O + Weight,
                   family = gaussian(link = "identity"),
                   mismodel = pi|qi ~ 1,
                   data = GeneRepeat$Main,validationdata = GeneRepeat$Validation,
                   err.var = "Totdist", mis.var = "Batch_O", err.true = "Tibia_T",
                   mis.true = "Batch_T", err.mat = NULL, repeated = TRUE,
                   repind= list(Totdist=c("Totdist5" , "Totdist10" , "Totdist15" ,
                                          "Totdist20",  "Totdist25" , "Totdist30")),
                   lambda = NULL, M = 10, B = 20, nBoot = 50, extrapolation="quadratic",
                   solvefun = "nleqslv", initial = c(0,0,0,0,0.01))
summary(example2)

# Call:
#   augSIMEX(mainformula = rs223979909 ~ Totdist + Batch_O + Weight,
#            mismodel = pi | qi ~ 1, family = gaussian(link = "identity"),
#            data = GeneRepeat$Main, validationdata = GeneRepeat$Validation,
#            err.var = "Totdist", mis.var = "Batch_O", err.true = "Tibia_T",
#            mis.true = "Batch_T", err.mat = NULL, repeated = TRUE,
#                repind = list(Totdist = c("Totdist5",
#           "Totdist10", "Totdist15", "Totdist20", "Totdist25", "Totdist30")),
#            lambda = NULL, M = 10, B = 20, nBoot = 50, extrapolation = "quadratic",
#            solvefun = "nleqslv", initial = c(0, 0, 0, 0, 0.01))
#
#
# The error-prone variable: Totdist
# The corresponding true variable: Tibia_T
# Number of simulations: 20
# Number of iterations in bootstrap: 50
#
#
# Coefficients:
#   Estimate Std. Error t value Pr(>|t|)
# (Intercept)  7.479e-01  2.717e-01   2.752  0.00592 **
#   Tibia_T     -4.265e-05  2.803e-05  -1.521  0.12819
```

```
# Weight      -5.465e-03  9.047e-03  -0.604  0.54577
# Batch_T      1.227e-01  9.853e-02   1.245  0.21304
# ---
#   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
#
# (Dispersion parameter for gaussian family taken to be NULL)
#
# Null deviance: 255.04  on 671  degrees of freedom
# Residual deviance: 270.87  on 668  degrees of freedom
# AIC: 1306.5



######################
## Section 7 Discussion              ##
Data <- DataGenerator(Seeds[1],0.5,"1D")

## The main function of correcting measurement error and misclassification
Re_cpp <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
               data=Data$Main,validationdata=Data$Validation, subset=NULL,
               err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z", err.mat = NULL,
               lambda=NULL, M=3, B=3, nBoot=3, extrapolation=c("both"))
Re_R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
               data=Data$Main,validationdata=Data$Validation, subset=NULL,
               err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z", err.mat = NULL,
               lambda=NULL, M=3, B=3, nBoot=3, extrapolation=c("both"),cppmethod = F)

speedcompare<-microbenchmark(
  cpp=cpp <- augSIMEX(mainformula = Y~Xstar+W+Zstar,
                      mismodel=pi|qi~1, family = binomial,
                      data=Data$Main,validationdata=Data$Validation, subset=NULL,
                       err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                       err.mat = NULL,  lambda=NULL, M=2, B=2, nBoot=2,
                       extrapolation=c("both")),
  R=R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
                       data=Data$Main,validationdata=Data$Validation, subset=NULL,
                       err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                       err.mat = NULL,
```

```
                    lambda=NULL, M=2, B=2, nBoot=2, extrapolation=c("both"),
                    cppmethod = F), times = 50)
if (require("ggplot2")) {
  autoplot(speedcompare)
}

## figure 2: speed and the simulation parameter
# c("M","B","nBoot","type","parameter","time")

timerecord<-NULL
Data <- DataGenerator(Seeds[1],0.5,"1D")

for (N in 1:200){
  for (i in c(5,10,15,20,25)){
    start <- proc.time()
    R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
                  data=Data$Main,validationdata=Data$Validation, subset=NULL,
                  err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                  err.mat = NULL,
                  lambda=NULL, M=i, B=5, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(i,5,5,"1D","M",end[1]))
    start <- proc.time()
    R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
                  data=Data$Main,validationdata=Data$Validation, subset=NULL,
                  err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                  err.mat = NULL,
                  lambda=NULL, M=5, B=i, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,i,5,"1D","B",end[1]))
    start <- proc.time()
    R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
                  data=Data$Main,validationdata=Data$Validation, subset=NULL,
                  err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                  err.mat = NULL,
                  lambda=NULL, M=5, B=5, nBoot=i, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,5,i,"1D","nBoot",end[1]))
  }
```

```
}

Data <- DataGenerator(Seeds[1],0.5,"2D")
meformula <- Xstar.X1|Xstar.X2~-1+X.X1|-1+X.X2
for (N in 1:200){
  for (i in c(5,10,15,20,25)){
    start <- proc.time()
    Re<-augSIMEX(mainformula = Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
                 pimodel=pi~X.X1+X.X2+W.W1+W.W2,
                 qimodel=qi~X.X1+X.X2+W.W1+W.W2, family = binomial,
                 data=Data$Main,meformula=meformula,validationdata=Data$Validation,
                 subset=NULL,
                 err.var=c("Xstar.X1","Xstar.X2"), mis.var="Zstar",
                 err.true=c("X.X1","X.X2"), mis.true="Z", err.mat = NULL,
                 lambda=NULL, M=i, B=5, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(i,5,5,"2D","M",end[1]))
    start <- proc.time()
    Re<-augSIMEX(mainformula = Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
                 pimodel=pi~X.X1+X.X2+W.W1+W.W2,
                 qimodel=qi~X.X1+X.X2+W.W1+W.W2, family = binomial,
                 data=Data$Main,meformula=meformula,validationdata=Data$Validation,
                 subset=NULL,
                 err.var=c("Xstar.X1","Xstar.X2"), mis.var="Zstar",
                 err.true=c("X.X1","X.X2"), mis.true="Z", err.mat = NULL,
                 lambda=NULL, M=5, B=i, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,i,5,"2D","B",end[1]))
    start <- proc.time()
    Re<-augSIMEX(mainformula = Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
                 pimodel=pi~X.X1+X.X2+W.W1+W.W2,
                 qimodel=qi~X.X1+X.X2+W.W1+W.W2, family = binomial,
                 data=Data$Main,meformula=meformula,validationdata=Data$Validation,
                 subset=NULL,
                 err.var=c("Xstar.X1","Xstar.X2"), mis.var="Zstar",
                 err.true=c("X.X1","X.X2"), mis.true="Z", err.mat = NULL,
                 lambda=NULL, M=5, B=5, nBoot=i, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,5,i,"2D","nBoot",end[1]))
```

```
  }
}

Data <- DataGenerator(Seeds[1],0.5,"RM")
for (N in 1:200){
  for (i in c(5,10,15,20,25)){
    start <- proc.time()
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=logit),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
                   lambda=NULL, M=i, B=5, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(i,5,5,"RM","M",end[1]))
    start <- proc.time()
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=logit),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
                   lambda=NULL, M=5, B=i, nBoot=5, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,i,5,"RM","B",end[1]))
    start <- proc.time()
    Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=logit),
                   data=Data$Main,validationdata=Data$Validation, subset=NULL,
                   err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                   err.mat = NULL,
                   repeated = TRUE,repind=list(Xstar1=c("Xstar1","Xstar2")),
                   lambda=NULL, M=5, B=5, nBoot=i, extrapolation=c("both"))
    end <- proc.time() - start
    timerecord<-rbind(timerecord,c(5,5,i,"RM","nBoot",end[1]))
  }
}

TimeSheet <- data.frame(M=as.numeric(timerecord[,1]),
                        B=as.numeric(timerecord[,2]),
                        nBoot=as.numeric(timerecord[,3]),
```

```
                        type=timerecord[,4],
                        parameter=timerecord[,5],
                        time=as.numeric(timerecord[,6]))


TimeSheetagg <- aggregate(TimeSheet$time,by=list(TimeSheet$M,TimeSheet$B,
                  TimeSheet$nBoot,TimeSheet$parameter,TimeSheet$type),FUN=mean)
TimeSheetagg <- data.frame(TimeSheetagg, choice = rep(c(5,10,15,20,25),9))
colnames(TimeSheetagg) <- c("M","B","T","parameter","study","time","choice")
# save(TimeSheetagg,file="TimeSheet.Rdata")
TimeSheetagg$parameter <- as.character(TimeSheetagg$parameter)
TimeSheetagg[TimeSheetagg$parameter=="nBoot","parameter"] <-
          rep("T",length(TimeSheetagg[TimeSheetagg$parameter=="nBoot","parameter"]))


g <- ggplot( TimeSheetagg , aes(x=choice, y= time, colour = study)) +
  geom_line(aes(linetype=study),size=0.8)+
  scale_colour_brewer(palette = "Set1") +
  scale_linetype_manual(values=c("twodash", "dotted","solid")) +
  facet_wrap(~parameter)+
  theme(text=element_text(size=13, family="mono"))

cbPalette <- c("#56B4E9","#999999", "#E69F00")

ggplotly(g+ scale_color_manual(values=cbPalette))



## figure 3: speed, sample size
timerecord2<-NULL
for (N in 1:100){
  for (i in c(300+0:12*100)){
    for (j in c(500+0:10*50)){
      Data <- DataGenerator2(Seeds[1],0.5,"1D",nm=i,nv=j,link="logit")
      start <- proc.time()
      R <- augSIMEX(mainformula = Y~Xstar+W+Zstar, mismodel=pi|qi~1, family = binomial,
                    data=Data$Main,validationdata=Data$Validation, subset=NULL,
                    err.var="Xstar", mis.var="Zstar", err.true="X", mis.true="Z",
                    err.mat = NULL,lambda=NULL, M=5, B=5, nBoot=5, extrapolation=c("both"))
```

```
      end <- proc.time() - start
      timerecord2<-rbind(timerecord2,c(i,j,"1D",end[1]))
      Data <- DataGenerator2(Seeds[1],0.5,"2D",nm=i,nv=j,link="logit")
      meformula <- Xstar.X1|Xstar.X2~-1+X.X1|-1+X.X2
      start <- proc.time()
      Re<-augSIMEX(mainformula = Y~Xstar.X1+Xstar.X2+Zstar+W.W1+W.W2,
                   pimodel=pi~X.X1+X.X2+W.W1+W.W2,
                   qimodel=qi~X.X1+X.X2+W.W1+W.W2, family = binomial,
                   data=Data$Main,meformula=meformula,validationdata=Data$Validation,
                   subset=NULL, err.var=c("Xstar.X1","Xstar.X2"), mis.var="Zstar",
                   err.true=c("X.X1","X.X2"), mis.true="Z", err.mat = NULL,
                   lambda=NULL, M=5, B=5, nBoot=5, extrapolation=c("both"))
      end <- proc.time() - start
      timerecord2<-rbind(timerecord2,c(i,j,"2D",end[1]))
      Data <- DataGenerator2(Seeds[1],0.5,"RM",nm=i,nv=j,link="logit")
      start <- proc.time()
      Re <- augSIMEX(mainformula = Y~Xstar1+Zstar+W, family = binomial(link=logit),
                     data=Data$Main,validationdata=Data$Validation, subset=NULL,
                     err.var="Xstar1", mis.var="Zstar", err.true="X", mis.true="Z",
                     err.mat = NULL, repeated = TRUE,
                     repind=list(Xstar1=c("Xstar1","Xstar2")),
                     lambda=NULL, M=5, B=5, nBoot=5, extrapolation=c("both"))
      end <- proc.time() - start
      timerecord2<-rbind(timerecord2,c(i,j,"RM",end[1]))
    }
  }
  cat("N=",N,"\n")
}

TimeSheet <- data.frame(nm=as.numeric(timerecord2[,1]),
                        nv=as.numeric(timerecord2[,2]),
                        type=timerecord2[,3],
                        time=as.numeric(timerecord2[,4]))

TimeSheetagg2 <- aggregate(TimeSheet$time,by=list(TimeSheet$nm,TimeSheet$nv,
                                                  TimeSheet$type),FUN=mean)
colnames(TimeSheetagg2) <- c("nm","nv","type","time")

# save(TimeSheetagg2,file="Sheetagg2.RData")
```

```
v <- ggplot(TimeSheetagg2, aes(nm, nv, z = time)) +
 facet_wrap(~type)+
  theme(text=element_text(size=13, family="mono"))+
   geom_raster(aes(fill = time)) +
   geom_contour(colour = "white")

ggplotly(v)

timerecord2<-matrix(timerecord2,nrow=7)
timerecord2<-rowMeans(timerecord2)
```