Package 'EHRmuse'

January 28, 2025

Type Package

Title Multi-Cohort Selection Bias Correction using IPW and AIPW Methods

Version 0.0.2.1

Description Comprehensive toolkit for addressing selection bias in binary disease models across diverse non-probability samples, each with unique selection mechanisms. It utilizes Inverse Probability Weighting (IPW) and Augmented Inverse Probability Weighting (AIPW) methods to reduce selection bias effectively in multiple non-probability cohorts by integrating data from either individual-level or summary-level external sources. The package also provides a variety of variance estimation techniques. Please refer to Kundu et al. <doi:10.48550/arXiv.2412.00228>.

License GPL (>= 2)

URL https://github.com/Ritoban1/EHRmuse

BugReports https://github.com/Ritoban1/EHRmuse/issues

Depends R (>= 4.0.0)

Imports dplyr (>= 1.0.0), magrittr, MASS, nleqslv (>= 3.3.2), xgboost (>= 1.4.1), survey (>= 4.1.0), stats, nnet (>= 7.3-17), simplexreg (>= 0.1.6)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Ritoban Kundu [aut], Michael Kleinsasser [cre]

Maintainer Michael Kleinsasser <biostat-cran-manager@umich.edu>

Repository CRAN

Date/Publication 2025-01-28 14:50:02 UTC

Contents

EHRmuse *IPW and AIPW Methods for Multi-cohort Selection Bias in Non-probability Samples*

Description

IPW and AIPW Methods for Multi-cohort Selection Bias in Non-probability Samples

Usage

```
EHRmuse(
  K,
  Z_names,
  intdata_list,
  N = NULL,
  UW_CS = FALSE,
  IPW = FALSE,
  weights_user = NULL,
  AIPW = FALSE,
  ipw_method = "PL",
  extdata = NULL,
  marginals_list = NULL,
  select_var_list = NULL,
  aux_var_list = NULL,
  Weights_e = NULL,
  aux_model = "XGBoost",
  variance = FALSE,
  type_var = "approx"
)
```

Arguments

K	Necessary Input. Number of cohorts. Should be a numeric positive integer.
Z_names	Necessary Input. A character vector containing the names of the Z variables or disease model covariates.
intdata_list	Necessary Input. A list of size K where each element of list corresponds to the data for each of the K multiple cohorts including the disease indicator D, the Z variables and the selection variables in the disease model. Each data should be of the form of a data frame. Please include a column named "id" to indicate unique identifiers to the units.
N	Target Population Size.
UW_CS	An indicator variable (TRUE or FALSE) for using the unweighted logistic regression model with cohort-specific intercepts.
IPW	An indicator variable (TRUE or FALSE) for using the Inverse Probability Weighted Methods (IPW) methods.

`weights_user` User specified weights. A numeric vector of weights for the combined data (not duplicated).

`AIPW` An indicator variable (TRUE or FALSE) for using the Joint Augmented IPW method. If AIPW is TRUE, please also input IPW to be TRUE.

`ipw_method` If IPW is TRUE, specify the IPW method to be used. A character variable. Default is PL (Pseudolikelihood). Other options are Simplex Regression (SR), Calibration (CL) or User Specified (US).

`extdata` If IPW method is set to PL or SR or AIPW is TRUE, please provide a data frame containing individual level external data which is a probability sample like NHANES. The external data should contain all the selection variables and if AIPW is TRUE, then also all the auxiliary score model variables. Please include a column named "id" to indicate unique identifiers to the units.

`marginals_list` If IPW method is set to CL, please provide a list of size K in which each element is a numeric vector containing the marginal sums of the selection variables of each of the K cohorts. Please ensure the first element for each of the K numeric vector should be the population size, N.

`select_var_list`
If IPW is set to be TRUE, please provide a list of size K in which each element is a character vector corresponding to the selection variables' names for each of the K cohorts.

`aux_var_list` If AIPW is set to be TRUE, please provide a list of size K in which each element is a character vector corresponding to the auxiliary score model variables' names for each of the K cohorts.

`Weights_e` If IPW method is set to PL or SR or AIPW is TRUE, please provide the known selection weights for the external probability sample. The input should be a numeric vector.

`aux_model` If AIPW is true, please provide the auxiliary score model. Default is XGboost.

`variance` An indicator variable (TRUE or FALSE) whether variance should be computed or not, along with the point estimate.

`type_var` If variance is true, indicate the method type to be used for computing the variance. For the unweighted method, do not provide any type. For IPW methods, PL and CL, we have two options, asy (asymptotic variance incorporating the variance from nuisance parameters) and "approx" ignoring the variance from nuisance parameters. For SR and AIPW, we have only the approx method. The default for IPW and AIPW methods is approx.

Value

If variance=TRUE, it will return a list of estimate vector and variance vector. If variance=FALSE, it will return an estimate vector.

Examples

```
#library(MASS)

K=3 ## Number of Cohorts
```

```
set.seed(100)
mean_w_p=0
mean_z_1=0
mean_z_2=0
mean_z_3=0
corr=0.5
var_z_w_p=matrix(c(1,corr,corr,corr,
                   corr,1,corr,corr,
                   corr,corr,1,corr,
                   corr,corr,corr,1),
                 nrow=4,ncol=4)

theta=c(-2,0.35,0.45,0.25) ## Theta_Z vector
N=5e4 ## Population size

### selection models
dw=1
dwz1=c(1,0.8,0.6)
dwz2=c(0.6,0.8,1)
dwz3=rep(1,3)

gamma_ext=c(-0.6,1.2,0.4,-0.2,0.5)
gamma_int_1=c(-1,1.5,0.2,0.8,-0.3)
gamma_int_2=c(-1,1.25,0.4,0.6)
gamma_int_3=c(-3,0.8,0.5)

## Generation of population level data
simu_popu<-function(N,mean_w_p,mean_z_1,mean_z_2,mean_z_3,
                    var_z_w_p,theta,dw){
   cov<- MASS::mvrnorm(n = N, mu = c(mean_w_p,mean_z_1,mean_z_2,mean_z_3), Sigma = var_z_w_p)
    data <- data.frame(Z1 = cov[, 2], Z2 = cov[, 3], Z3=cov[,4])
    W_p=cov[,1]
    # Generate random uniforms
    #set.seed(5678)
    U1 <- runif(N)
    #set.seed(4321)
    # Generate Disease Status
    DISEASE <- expit(theta[1] + theta[2] * data$Z1 + theta[3]*data$Z2 +theta[4]*data$Z3)
    data$D   <- ifelse(DISEASE > U1, 1, 0)
    # Relate W_p and D
    data$W_1 <- W_p + dw* data$D + dwz1[1]*data$Z1 +
        dwz2[1]*data$Z2 + dwz3[1]*data$Z3 +
        rnorm(n=N,0,1)

    data$W_2 <- W_p + dw* data$D + dwz1[2]*data$Z1 +
        dwz2[2]*data$Z2 + dwz3[2]*data$Z3 +
        rnorm(n=N,0,1)

    data$W_3 <- W_p + dw* data$D + dwz1[3]*data$Z1 +
        dwz2[3]*data$Z2 + dwz3[3]*data$Z3 +
        rnorm(n=N,0,1)

    data$id=c(1:N)
```

```
    return(data)
}
## Generation of external individual level data
simu_ext<-function(data,gamma_ext){
    U2e <- runif(N)
    # Generate Sampling Status
    SELECT <-0.75*expit(gamma_ext[1] +
                            gamma_ext[2]* data$D +
                            gamma_ext[3] * data$Z1 +
                            gamma_ext[4]* data$Z2 +
                            gamma_ext[5] * data$Z3)
    S_e  <- ifelse(SELECT > U2e, TRUE, FALSE)
    # Observed Data
    data_e <- data[which(S_e==1),]
    data_e$Select_Weights = 0.75*expit(gamma_ext[1] +
                                           gamma_ext[2]* data_e$D +
                                           gamma_ext[3] * data_e$Z1 +
                                           gamma_ext[4]* data_e$Z2 +
                                           gamma_ext[5] * data_e$Z3)
    return(data_e)
}
## Generation of internal data 1
simu_int_1<-function(data,gamma_int_1){
    U2i <- runif(N)
    # Generate Sampling Status
    SELECT <- expit(cbind(1,data$D,data$W_1,data$Z2,data$Z3)
                    %*% gamma_int_1)
    S_i  <- ifelse(SELECT > U2i, TRUE, FALSE)
    # Observed Data
    data_i <- data[which(S_i==1),]
    return(data_i)
}

## Generation of internal data 2
simu_int_2<-function(data,gamma_int_2){
    U2i <- runif(N)
    # Generate Sampling Status
    SELECT <- expit(cbind(1,data$D,data$W_2,data$Z3)
                    %*% gamma_int_2)
    S_i  <- ifelse(SELECT > U2i, TRUE, FALSE)
    # Observed Data
    data_i <- data[which(S_i==1),]
    return(data_i)
}


## Generation of internal data 3
simu_int_3<-function(data,gamma_int_3){
    U2i <- runif(N)
    # Generate Sampling Status
    SELECT <- expit(cbind(1,data$W_3,data$Z2)
                    %*% gamma_int_3)
    S_i  <- ifelse(SELECT > U2i, TRUE, FALSE)
```

```
    # Observed Data
    data_i <- data[which(S_i==1),]
    return(data_i)
}

data=simu_popu(N,mean_w_p,mean_z_1,mean_z_2,mean_z_3,
               var_z_w_p,theta,dw)

extdata=simu_ext(data,gamma_ext)


intdata1=simu_int_1(data,gamma_int_1)
intdata2=simu_int_2(data,gamma_int_2)
intdata3=simu_int_3(data,gamma_int_3)

## names of selection variables in each cohort
select_var_list=list(c("D","W_1","Z2","Z3"),c("D","W_2","Z3"),c("W_3","Z2"))

## names of auxiliary variables in each cohort
aux_var_list=list(c("D","W_1","Z2","Z3"),c("D","W_2","Z3"),c("W_3","Z2"))

## list of internal data
intdata_list=list(intdata1,intdata2,intdata3)
## names of Z variables
Z_names=c("Z1","Z2","Z3")

theta ## actual theta_z
res_uw=EHRmuse(K=K,N=N,Z_names=Z_names,
               intdata_list=intdata_list,variance = TRUE)
```

`expit` *Expit*

Description

Expit

Usage

```
expit(x)
```

Arguments

`x`	numeric vector

Value

exp(x)/(1+exp(x))

Examples

```
expit(1)
```

Index