Supplementary

Table S1 Estimating β_1 for case control design at 1.95% prevalence and non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	0.010	0.153	0.151	95.0
100	90	0.011	0.155	0.150	95.2
100	80	−0.009	0.151	0.150	95.8
99.9	100	−0.075	0.140	0.148	93.6
99.9	90	−0.081	0.152	0.147	90.8
99.9	80	−0.099	0.145	0.146	90.4
99.5	100	−0.271	0.149	0.139	49.6
99.5	90	−0.296	0.136	0.139	45.2
99.5	80	−0.322	0.139	0.139	37.4
99	100	−0.422	0.137	0.135	11.2
99	90	−0.442	0.130	0.135	8.4
99	80	−0.477	0.138	0.134	6.0
98.5	100	−0.526	0.133	0.133	2.8
98.5	90	−0.540	0.131	0.132	1.4
98.5	80	−0.558	0.128	0.132	0.8
98	100	−0.571	0.133	0.132	1.4
98	90	−0.611	0.126	0.131	0.4
98	80	−0.626	0.127	0.131	0.0

https://dx.doi.org/10.21037/ace-2025-3

Table S2 Estimating β_1 for case control design at 7.5% prevalence and non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	−0.003	0.149	0.149	94.8
100	90	−0.013	0.154	0.148	93.8
100	80	−0.012	0.146	0.148	95.2
99.9	100	−0.015	0.147	0.148	95.4
99.9	90	−0.030	0.146	0.147	95.2
99.9	80	−0.040	0.142	0.147	95.4
99.5	100	−0.090	0.154	0.145	89.4
99.5	90	−0.118	0.148	0.144	85.4
99.5	80	−0.133	0.152	0.144	84.4
99	100	−0.159	0.142	0.142	82.4
99	90	−0.190	0.140	0.142	72.6
99	80	−0.215	0.144	0.141	66.4
98.5	100	−0.228	0.139	0.141	62.6
98.5	90	−0.252	0.137	0.140	57.2
98.5	80	−0.282	0.138	0.139	48.4
98	100	−0.262	0.142	0.139	54.6
98	90	−0.300	0.141	0.138	39.2
98	80	−0.329	0.130	0.137	34.0

https://dx.doi.org/10.21037/ace-2025-3

Table S3 Estimating β_2 for case control design at 1.95% prevalence and non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	0.010	0.088	0.081	93.0
100	90	–0.001	0.081	0.081	95.6
100	80	–0.004	0.084	0.081	94.0
99.9	100	–0.067	0.079	0.078	83.6
99.9	90	–0.082	0.081	0.078	79.2
99.9	80	–0.095	0.081	0.077	74.2
99.5	100	–0.272	0.070	0.071	2.6
99.5	90	–0.296	0.073	0.070	3.0
99.5	80	–0.315	0.069	0.070	0.8
99	100	–0.416	0.064	0.067	0.0
99	90	–0.438	0.064	0.067	0.0
99	80	–0.467	0.063	0.066	0.0
98.5	100	–0.502	0.065	0.065	0.0
98.5	90	–0.527	0.062	0.065	0.0
98.5	80	–0.558	0.064	0.064	0.0
98	100	–0.564	0.065	0.064	0.0
98	90	–0.592	0.063	0.064	0.0
98	80	–0.618	0.063	0.063	0.0

https://dx.doi.org/10.21037/ace-2025-3

Table S4 Estimating β_2 for case control design at 7.5% prevalence and non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	0.006	0.084	0.082	94.2
100	90	–0.008	0.077	0.082	96.8
100	80	–0.014	0.080	0.081	94.8
99.9	100	–0.007	0.083	0.082	94.4
99.9	90	–0.034	0.080	0.081	92.6
99.9	80	–0.042	0.080	0.080	91.8
99.5	100	–0.084	0.080	0.079	77.2
99.5	90	–0.104	0.084	0.078	67.8
99.5	80	–0.128	0.078	0.077	60.0
99	100	–0.161	0.076	0.076	44.2
99	90	–0.184	0.072	0.075	32.4
99	80	–0.207	0.075	0.075	22.6
98.5	100	–0.212	0.076	0.074	19.4
98.5	90	–0.242	0.072	0.073	10.4
98.5	80	–0.271	0.068	0.072	4.8
98	100	–0.263	0.072	0.073	6.0
98	90	–0.299	0.077	0.072	2.0
98	80	–0.328	0.074	0.071	0.6

https://dx.doi.org/10.21037/ace-2025-3

Table S5 Case control design comparison at 100% specificity

XXXXXXX	Non-differential		Differential			
			XXXXXXX		XXXXXXX	
	β_1	β_2	β_1	β_2	β_1	β_2
Sensitivity	$Se_0 = Se_1 = 0.80$		$Se_0 = 0.90$ $Se_1 = 0.757$		$Se_0 = 0.757$ $Se_1 = 0.90$	
Point estimate	0.988	0.986	0.786	0.983	1.156	0.979
Bias	–0.012	–0.014	–0.214	–0.017	0.156	–0.021
Coverage (%)	95.20	94.80	66.80	93.60	82.40	92.60
Standard deviation	0.146	0.080	0.142	0.085	0.150	0.085
Mean of standard errors	0.148	0.081	0.145	0.081	0.151	0.082

Table S6 Case control design comparison at 100% sensitivity

XXXXXX	Non-differential		Differential			
			XXXXXX		XXXXXX	
	β_1	β_2	β_1	β_2	β_1	β_2
Specificity	$\boldsymbol{Sp_0 = Sp_1 = 0.995}$		$\boldsymbol{Sp_0 = 1.00}$ $\boldsymbol{Sp_1 = 0.99}$		$\boldsymbol{Sp_0 = 0.99}$ $\boldsymbol{Sp_1 = 1.00}$	
Point estimate	0.911	0.909	1.115	0.934	0.716	0.9
Bias	–0.089	–0.091	0.115	–0.066	–0.284	–0.1
Coverage (%)	89.60	78.80	87.60	86.20	46.60	71.40
Standard deviation	0.148	0.078	0.150	0.077	0.139	0.080
Mean of standard errors	0.145	0.078	0.149	0.080	0.143	0.078

https://dx.doi.org/10.21037/ace-2025-3

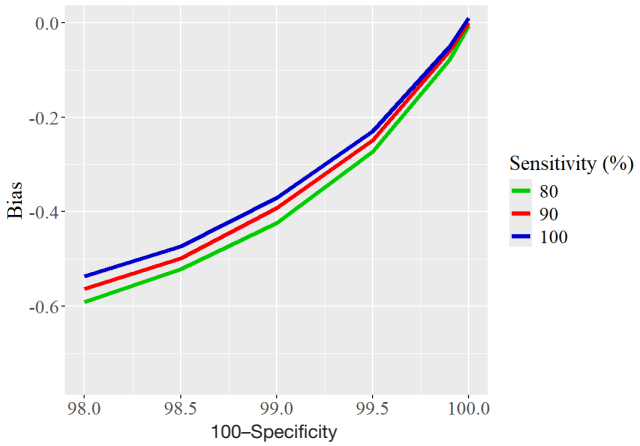

Figure S1 Bias from estimating β_1 at 1.95% prevalence for cohort design with non-differential misclassification becomes worse as sensitivity and specificity decrease.

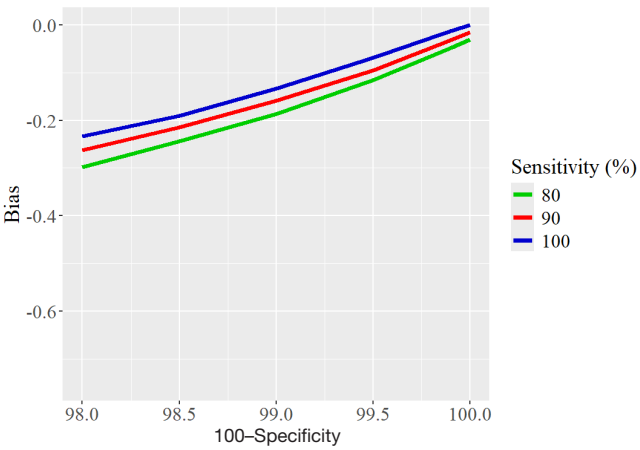

Figure S2 Bias from estimating β_1 at 7.5% prevalence with non-differential misclassification for cohort design is lower compared to bias obtained at 1.95% prevalence.

https://dx.doi.org/10.21037/ace-2025-3

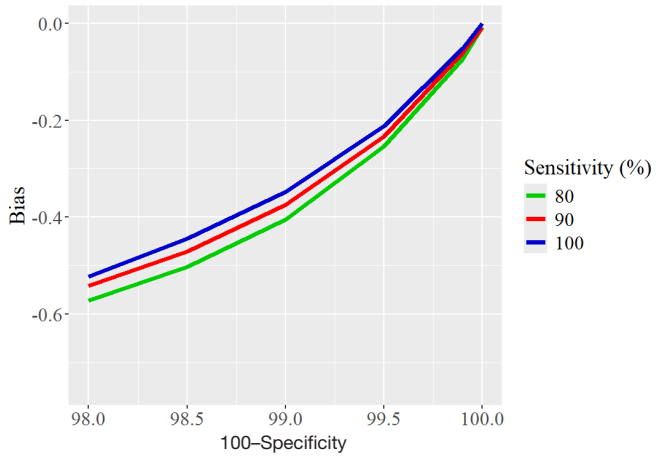

Figure S3 Cohort design bias from estimating β_2 with non-differential misclassification at 1.95% prevalence worsens as misclassification rates increase.

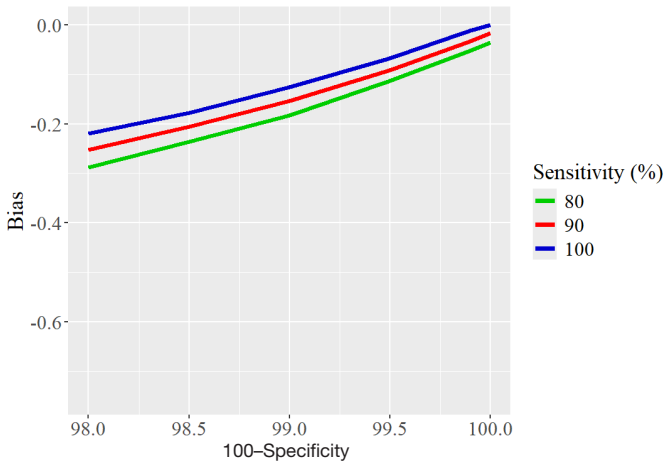

Figure S4 Bias from estimating β_2 of cohort design with non-differential misclassification at 7.5% prevalence is lower than bias obtained at lower cancer prevalence.

https://dx.doi.org/10.21037/ace-2025-3

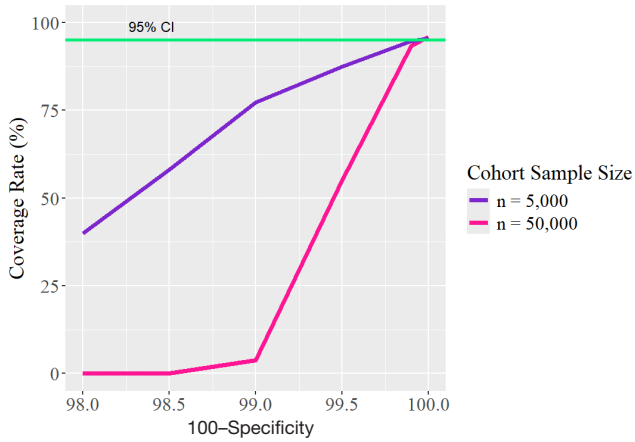

Figure S5 Coverage rates of β_1 from cohort design with 5,000 sample size at 1.95% prevalence and 100% sensitivity decreases at slower rate than 50,000 sample size.

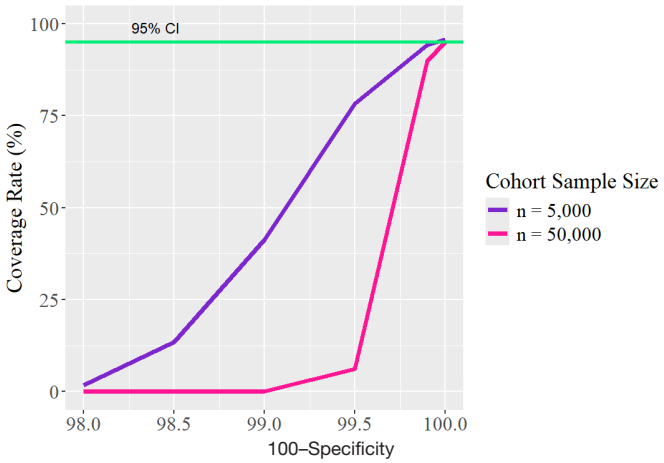

Figure S6 Coverage rates of β_2 from cohort design with 50,000 sample size at 7.5% prevalence and 100% sensitivity decrease more rapidly than cohort design with 5,000 sample size.

https://dx.doi.org/10.21037/ace-2025-3

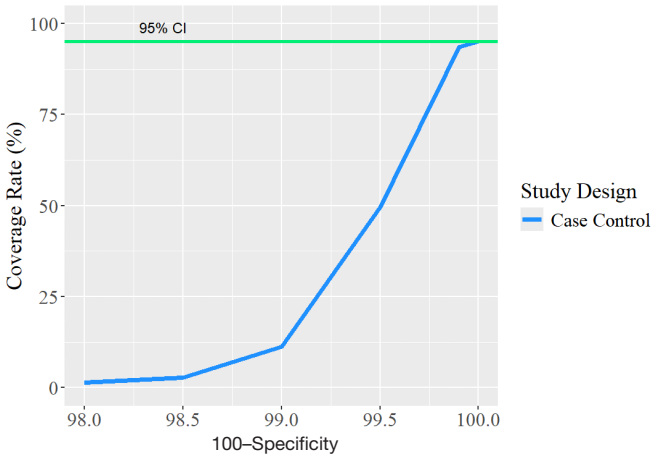

Figure S7 Coverage rates of β_1 from case control design at 1.95% prevalence and 100% sensitivity decrease rapidly as specificity decreases.

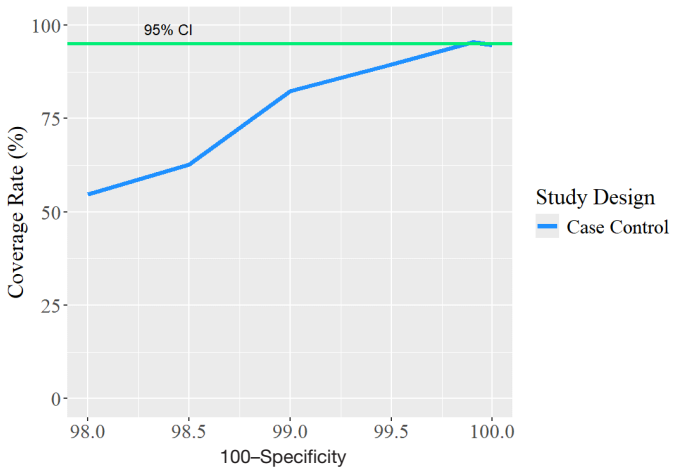

Figure S8 Coverage rates of β_1 from case control design at 7.5% prevalence and 100% sensitivity decrease at a slower rate.

https://dx.doi.org/10.21037/ace-2025-3

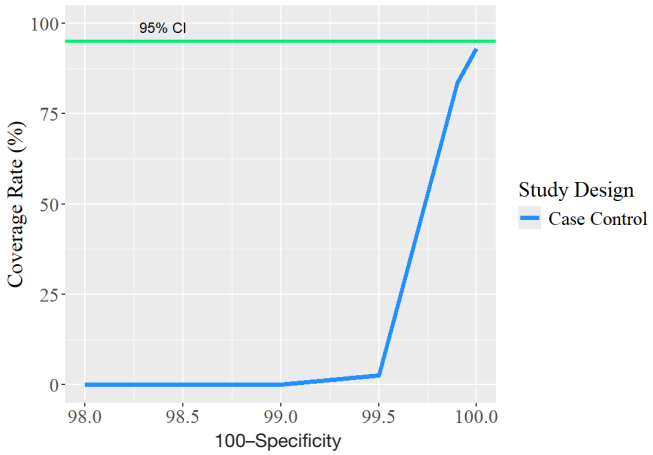

Figure S9 At specificity decreases, coverage rates of β_2 for case control design at 1.95% prevalence and 100% sensitivity rapidly decreases.

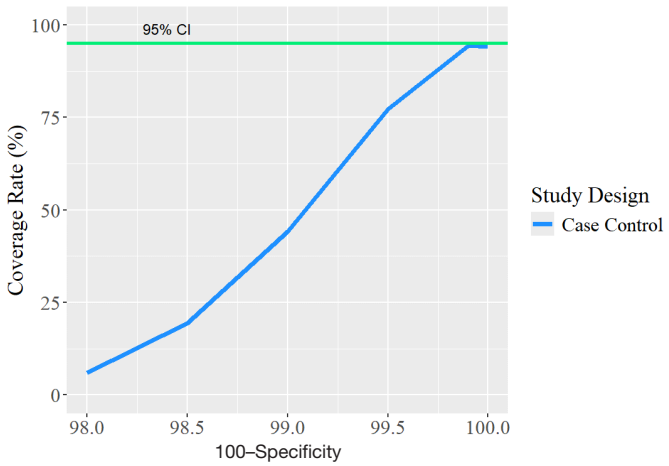

Figure S10 As specificity decreases, coverage rates of β_2 for case control design at 7.5% prevalence and 100% sensitivity decrease.

https://dx.doi.org/10.21037/ace-2025-3

We performed a theoretical calculation to examine the impact of outcome misclassification in the estimation of odds ratios (ORs). Unlike the simulations where there are two covariates, we consider a simpler case here with only one binary covariate so we can analyze the 2×2 table. With the true outcome Y and binary covariate X, suppose the expected cell counts are a, b, c and d (*Table S5*). When the outcome is misclassified with the following subject-specific sensitivities and specificities:

$Se_x \equiv P(Y^* = 1|Y = 1, X = x)$

$Sp_x \equiv P(Y^* = 0|Y = 0, X = x),$

where $x = 0$ or 1, the formulas for expected cell counts (a^*, b^*, c^* and d^*) in the 2×2 table of Y^* and X are given in *Table S5*. Furthermore, if one performs a case-control sampling of the individuals, say, sampling the cases ($Y^* = 1$) with probability π_1 and the controls ($Y^* = 0$) with probability π_0. The expected cell counts become $\pi_0 a^*$, $\pi_1 b^*$, $\pi_0 c^*$ and $\pi_1 d^*$, and the resulting marginal OR between Y^* and X would remain the same ($\frac{a^* d^*}{b^* c^*}$) as the sampling probabilities cancel out. This explains why the expected OR estimates have similar bias in the cohort study and case-control study.

Moreover, we calculated the expected OR estimates over a series of settings of Se and Sp. We start with setting $a = 24{,}741$, $b = 270$, $c = 24{,}285$, and $d = 704$, corresponding to one of the simulation designs with the prevalence being 1.95%, and the sample size of 50,000. The true marginal OR between Y and X is 2.7. Then under each setting of outcome misclassification, the formulas in *Table S5* are used to calculate a^*, b^*, c^*, d^* and the expected OR estimation:

$$OR^* = \frac{a^* d^*}{b^* c^*}$$

When the misclassification is non-differential, all the estimated ORs are biased towards the null across the range of Se and Sp considered (*Figure S6*), and the bias is much more impacted by Sp than Se. For differential Se and Sp, *Figures S7,S8* show substantial bias in the estimated ORs that can be in either direction, similar to what we have seen in the simulation studies.

https://dx.doi.org/10.21037/ace-2025-3

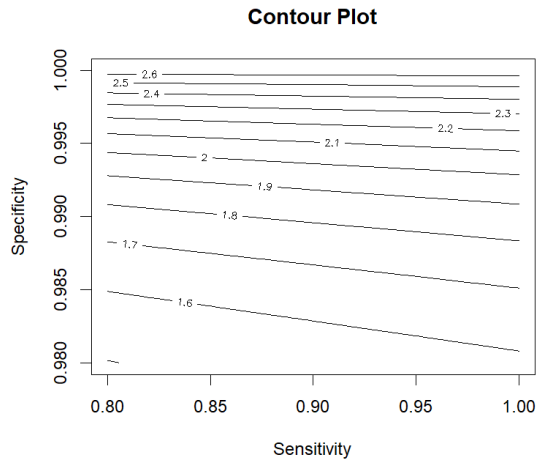

Figure S11 Contour plot of non-differential misclassification ORs where non-misclassified OR is 2.7.

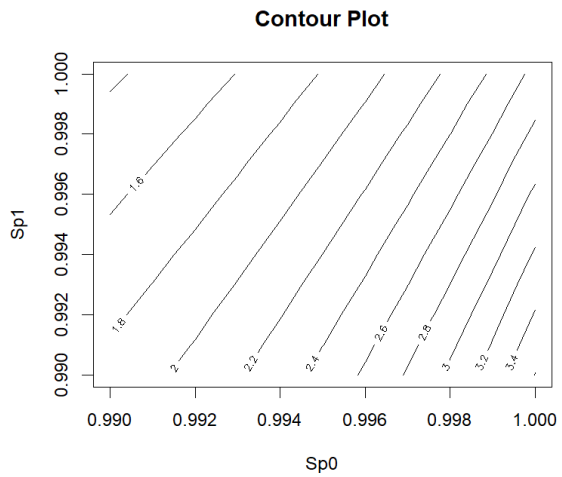

Figure S12 Contour plot of differential specificity levels resulting in similar ORs.

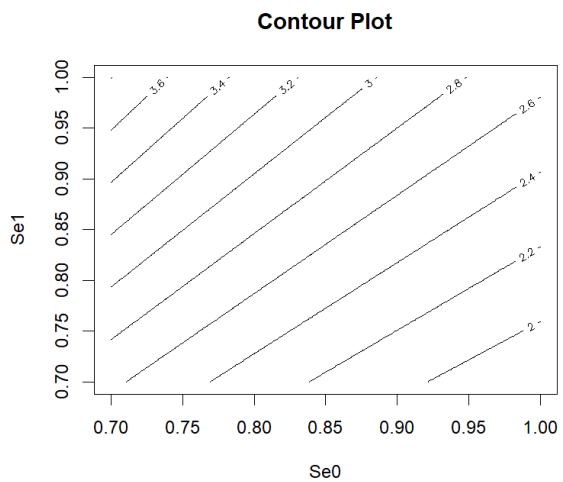

Figure S13 Contour plot of differential sensitivity levels resulting in similar ORs.

https://dx.doi.org/10.21037/ace-2025-3

Table S7 Asymptotic true outcome table

	$\boldsymbol{Y=0}$	$\boldsymbol{Y=0}$
$\boldsymbol{X=0}$	a	b
$\boldsymbol{X=1}$	c	d

Table S8 Asymptotic misclassified outcome table

	$\boldsymbol{Y^*=0}$	$\boldsymbol{Y^*=1}$
$\boldsymbol{X=0}$	$a^* = a \times Sp_0 + b \times (1 - Se_0)$	$b^* = a \times (1 - Sp_0) + b \times Se_0$
$\boldsymbol{X=1}$	$c^* = c \times Sp_1 + d \times (1 - Se_1)$	$d^* = c \times (1 - Sp_1) + d \times Se_1$

Table S9 Estimating β_2 for cohort design at 1.95% prevalence with non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	0.000	0.034	0.035	97.2
100	90	–0.009	0.037	0.036	93.8
100	80	–0.008	0.038	0.038	95.8
99.9	100	–0.052	0.035	0.033	64.2
99.9	90	–0.061	0.038	0.035	58.8
99.9	80	–0.073	0.040	0.037	45.0
99.5	100	–0.212	0.033	0.030	0.0
99.5	90	–0.233	0.032	0.031	0.0
99.5	80	–0.254	0.035	0.033	0.0
99	100	–0.347	0.031	0.027	0.0
99	90	–0.374	0.030	0.028	0.0
99	80	–0.405	0.031	0.029	0.0
98.5	100	–0.444	0.027	0.025	0.0
98.5	90	–0.471	0.028	0.026	0.0
98.5	80	–0.502	0.028	0.027	0.0
98	100	–0.513	0.025	0.024	0.0
98	90	–0.542	0.026	0.024	0.0
98	80	–0.572	0.027	0.025	0.0

https://dx.doi.org/10.21037/ace-2025-3

Table S10 Estimating β_2 of cohort design at 7.5% prevalence with non-differential misclassification

Specificity (%)	Sensitivity (%)	Bias	Standard deviation	Mean of standard errors	Coverage (%)
100	100	0.000	0.020	0.020	95.0
100	90	–0.017	0.020	0.020	88.0
100	80	–0.035	0.021	0.021	63.2
99.9	100	–0.011	0.020	0.020	89.8
99.9	90	–0.033	0.020	0.020	64.6
99.9	80	–0.052	0.021	0.021	30.0
99.5	100	–0.067	0.020	0.019	6.2
99.5	90	–0.091	0.021	0.019	0.2
99.5	80	–0.113	0.021	0.020	0.0
99	100	–0.126	0.019	0.018	0.0
99	90	–0.154	0.019	0.019	0.0
99	80	–0.183	0.020	0.019	0.0
98.5	100	–0.177	0.017	0.018	0.0
98.5	90	–0.205	0.019	0.018	0.0
98.5	80	–0.236	0.019	0.019	0.0
98	100	–0.22.0	0.017	0.017	0.0
98	90	–0.253	0.019	0.017	0.0
98	80	–0.288	0.019	0.018	0.0

https://dx.doi.org/10.21037/ace-2025-3

The R code used for conducting the simulations is provided below with comments to describe the purpose of any functions or variables used.

Important Functions:

#This function generates a dataset containing the true cancer outcome (Y) and misclassified cancer outcome (Y^*) based on non-differential misclassification. This is meant to simulation a population in a cohort study.

```
# Function parameters:
#   col_num = 7: defines the number of columns in population dataframe; this is a fixed value for this function
#   row_num: the number of individuals in population
#   true_bt_vect = (β0,β1,β2) = (β0,1,1): the beta parameters for simulation scenario
#   specificity: decimal representing Sp
#   sensitivity: decimal representing Se
const_sens_spec_data_gen <- function(col_num, row_num, true_bt_vect, specificity, sensitivity){
  df <- data.frame(matrix(ncol = col_num, nrow = row_num))
  colnames(df) <- c("X1","X2","P1","Y","Z0","Z1","Ystar")

  df[,1] <- rbinom(row_num, 1, 0.5) #X1: binary
  df[,2] <- rnorm(row_num, 0, 1) #X2: continuous

  logit1 <- df[,2]*true_bt_vect[3]+df[,1]*true_bt_vect[2]+true_bt_vect[1]
  prob1 <- exp(logit1) / (1 + exp(logit1))
  df[,3] <- prob1 #P1
  df[,4] <- rbinom(row_num, 1, prob1) #Y - true Y's

  # redefining Y* =>  simulating linkage mess ups manually
  df[,5] <- rbinom(row_num, 1, specificity) #Z0 : binary (Z0 controls Y* outcomes when Y=0)
  df[,6] <- rbinom(row_num, 1, sensitivity) #Z1 : binary (Z1 controls Y* outcomes when Y=1)

  #Possible Outcomes:
  # If Z0=0 and Y=0, then Y*=1 (false positive)
  # If Z0=1 and Y=0, then Y*=0 (true negative)
  # If Z1=0 and Y=1, then Y*=0 (false negative)
  # If Z1=1 and Y=1, then Y*=1 (true positive)

  df[,7] = df[,4] #set Y* = Y
  df[,7][df[,4]==1] = df[,6][df[,4]==1] # When Y=1, define Y*=Z1
  df[,7][df[,4]==0] = 1-df[,5][df[,4]==0] # When Y=0, define Y*=1-Z0 (aka the flipped Z0 [fs Z0=1, then 1-Z0=0])

  return(df)
}

# This function is used to simulate Y^* for differential misclassification:
  # Outcomes for Y^*
  # Y=0 -> Ys=1 with prob (1-Sp)
  # Y=0 -> Ys=0 with prob Sp
  # Y=1 -> Ys=0 with prob (1-Se)
```


https://dx.doi.org/10.21037/ace-2025-3

```
  # Y=1 -> Ys=1 with prob Se

#   Parameter meanings:
# Y_vect: vector of true cancer outcomes (Y)
# Sens_setting: Se
# Spec_setting: Sp

generating_Ystar <- function(Y_vect, sens_setting, spec_setting){
  Ystar = Y_vect

  n_Y_equals0 <- sum(Y_vect ==0)
  n_Y_equals1 <- sum(Y_vect ==1)

  Ystar[Y_vect==1] <- rbinom(n_Y_equals1, 1, sens_setting)
  Ystar[Y_vect==0] <- rbinom(n_Y_equals0, 1, 1-spec_setting)
  return(Ystar)
}

# This function shows a shorter way to get a population dataset. It was used in the simulations for differential misclassification.

# row_num = population size
basic_data_gen <-function(row_num, true_betas_vect){

  df <- data.frame(matrix(ncol = 3, nrow = row_num))
  colnames(df) <- c("X1", "X2", "Y")

  df[,1] <- rbinom(row_num, 1, 0.5) #X1: binary
  df[,2] <- rnorm(row_num, 0, 1) #X2: continuous

  logit1 <- true_betas_vect[1] + df[,1]*true_betas_vect[2] + df[,2]*true_betas_vect[3]
  prob <- exp(logit1) / (1 + exp(logit1)) #P1
  df[,3] <- rbinom(row_num, 1, prob) #Y

  return(df)
}

# The following function was used to generate a sample for a case-control study.
sample_gen <- function(population, samp_size){

  # collects indexes of people who have cancer outcome of 0/1
  indexes_1 <- which(population$Ystar==1)
  indexes_0 <- which(population$Ystar==0)

  # randomly select people who have cancer and those who do not to create a sample
  select_1 <- sample(x=indexes_1, size=samp_size, replace=FALSE)
  select_0 <- sample(x=indexes_0, size=samp_size, replace=FALSE)

  # Information on Participants who are determined as diseased
```

 https://dx.doi.org/10.21037/ace-2025-3

```
  Y_star_pos_selected <- population$Ystar[c(select_1)]
  pos_x1_info <- population$X1[c(select_1)]
  pos_x2_info <- population$X2[c(select_1)]

  # Information on Participants who are determined as not diseased
  Y_star_neg_selected <- population$Ystar[c(select_0)]
  neg_x1_info <- population$X1[c(select_0)]
  neg_x2_info <- population$X2[c(select_0)]

  # combining all information for each variable together (x1,x2,y**)
  all_Y_star <- c(Y_star_pos_selected, Y_star_neg_selected)
  all_x1 <- c(pos_x1_info, neg_x1_info)
  all_x2 <- c(pos_x2_info, neg_x2_info)

  # creating sample with participants of both imperfect cancer outcomes (0/1)
  sample <- data.frame(matrix(ncol = 3, nrow = (samp_size*2)))
  sample[,1] <- all_Y_star
  sample[,2] <- all_x1
  sample[,3] <- all_x2

  colnames(sample) <- c("Ystarstar", "X1", "X2")

  return(sample)
}

# Function to run logistic regression for case control study:
retro_logit <- function(test_sample){
  # test_sample = a dataframe
  analysis <- glm(test_sample[,1] ~ (test_sample[,2]+test_sample[,3]), family="binomial", data = test_sample)
  summary(analysis)

  results <- summary(analysis)$coef
  return(results)
}

# Function for running logistic regression of cohort study:
new_cal_glm <- function(pop_dataset, c){
  # c = the column of the cancer outcome = 7, fixed value for function
  fit <- glm(pop_dataset[,c] ~ (pop_dataset[,1]+pop_dataset[,2]), family = "binomial", data = pop_dataset)
  summary(fit)

  coefs <- summary(fit)$coef
  return(coefs)
}

# Function for creating a blank matrix to store results from simulations:
blank_mtrx <- function(n_sims, n_cols, col_name_vect){
  # n_sims = number of simulations
```


https://dx.doi.org/10.21037/ace-2025-3

```
  # n_cols = number of beta coefficients = 3, fixed value throughout code
  df_blank <- data.frame(matrix(nrow=n_sims, ncol=n_cols))
  colnames(df_blank) <- c(col_name_vect)
  return(df_blank)
}

# Function to find coverage rates:
find_coverage_rates <- function(point_estimates_mtrx, std_err_mtrx, betas_vec, beta_index, cnf_lvl, sim_num){
  # point_estimates_mtrx = matrix of point estimates obtained from logistic regression
  # std_err_mtrx = matrix of standard errors obtained from logistic regression
  # betas_vec = beta coefficients
  # beta_index = index of beta coefficient to find coverage rate of
  # cnf_lvl = confidence level = 95%
  # sim_num = number of simulations run

  margin_err <- cnf_lvl*std_err_mtrx
  bias_dist <- abs(sweep(point_estimates_mtrx, MARGIN=2, STATS=betas_vec, FUN="-"))

  test_pe <- ifelse(bias_dist <= margin_err, TRUE, FALSE)

  t.count <- length(which(test_pe[,beta_index] == "TRUE"))
  f.count <- length(which(test_pe[,beta_index] == "FALSE"))

  cov_rate <- (t.count/ sim_num) * 100
  return(cov_rate)
}

# The following function calculates the differential Se or Sp for X_1=1. This calculation depends on inputting a specified overall Se or Sp and the misclassification for differential misclassification (Se or Sp) for participants with X_1=0.
# input definitions:
# - overall_sens and overall_spec: specified decimals for overall Se or Sp, respectively.
# - given_S0 and given_C0: specified decimals for Se or Sp, respectively, when X_1=0
# # NOTE: C values = Specificity , S values = Sensitivity

calc_sens_spec_x1 <- function(population_simulated, given_S0, given_C0, overall_sens, overall_spec){
  count_mtrx <- table(population_simulated$X1, population_simulated$Y)
  p <- count_mtrx[1,1] / (count_mtrx[1,1] + count_mtrx[2,1])
  pi_val <- count_mtrx[1,2] / (count_mtrx[1,2] + count_mtrx[2,2])

  S1 <- (overall_sens - (given_S0*pi_val)) / (1-pi_val)
  C1 <- (overall_spec - (given_C0*p)) / (1-p)

  return(c(given_S0, S1, given_C0, C1))
}
# The following function calculates an estimated Se and Sp from Y and Y*
testfun <- function(Y,Ys){
  ## input: Y and Y* = Ys
  # output: estimated Se and Sp
```


https://dx.doi.org/10.21037/ace-2025-3

```
  Se.est=mean(Ys[Y==1])       # P(Y*=1|Y=1)
  Sp.est=1-mean(Ys[Y==0])     # 1-P(Y*=1|Y=0)
  c(Se=Se.est,Sp=Sp.est)
}

# Simulation Code
# The following code runs the 500 simulations for a cohort and case-control study with non-differential misclassification:

# Global Variables
population_size <- 50000 # cohort sample size
new_sim_num <- 500 # number of simulations
sample_size <- 500 # case control size for each binary cancer outcome
betas <- c(-3.5,1,1)


# Making Matrix containing all parameters
prevs_beta0 <- c(rep(x=-5,times=18), rep(-4.5,18), rep(-4,18), rep(-3.5,18))
specs_vec <- c(rep(x=c(rep(1.0, 3), rep(0.999,3), rep(0.995,3), rep(0.99,3), rep(0.985,3), rep(0.98,3)),4))
sens_vec <- c(rep(x=seq(1.0,0.8,-0.1),times=24))

# 4x6x3 = 72 total combinations
parameters_mtrx <- matrix(NA,nrow=72,ncol=3)
parameters_mtrx[,1] <- prevs_beta0
parameters_mtrx[,2] <- specs_vec
parameters_mtrx[,3] <- sens_vec
colnames(parameters_mtrx) <- c("Prev Coeff", "Specs", "Sens")

# Defining Lists to Store Information
pops_saved <- NULL
raw_prevalences_stored = prevalences <- NULL
new_pro_PE.RESULT = new_pro_SE.RESULT = new_ret_PE.RESULT = new_ret_SE.RESULT <- NULL
new_sd_pro_df = new_sd_ret_df <- NULL

new_pro_PE_avg_df = new_ret_PE_avg_df <- NULL
new_pro_SE_avg_df = new_ret_SE_avg_df <- NULL
new_pro_b1_cov = new_pro_b2_cov <- NULL
new_ret_b1_cov = new_ret_b2_cov <- NULL

for (i in 1:nrow(parameters_mtrx)){
  pro_pe_df = pro_se_df = ret_pe_df = ret_se_df <- blank_mtrx(n_sims = new_sim_num,
                                          n_col = 3,
                                          c("Beta0", "Beta1", "Beta2"))
  betas[1] <- parameters_mtrx[i,1]
  cat("\n Betas:", betas)

  spec <- parameters_mtrx[i,2]
  cat("\n Specificity:", spec)
```


https://dx.doi.org/10.21037/ace-2025-3

```
  sens <- parameters_mtrx[i,3]
  cat("\n Sensitivity:", sens)

  set.seed(i+100)
  for (j in 1:new_sim_num){ # new_sim_num = 500 simulations

    cohort <- const_sens_spec_data_gen(col_num = 7,
                                       row_num = population_size,
                                       true_bt_vect = betas,
                                       specificity = spec,
                                       sensitivity = sens)
    # saves prevalences of dataset from each simulation
    raw_prevalences_stored[j] <- mean(cohort$Y)

    sample_col <- sample_gen(population = cohort,
                             samp_size = sample_size)

    # run logistic with prospective analysis
    prospec_coefs <- new_cal_glm(pop_dataset = cohort, c = 7)

    pro_pe_df[j, ] <- prospec_coefs[,1]
    pro_se_df[j, ] <- prospec_coefs[,2]

    # run logistic with retrospective analysis
    retro_coefs <- retro_logit(test_sample = sample_col)

    ret_pe_df[j, ] <- retro_coefs[,1]
    ret_se_df[j, ] <- retro_coefs[,2]

  }
  pops_saved[[i]] <- cohort
# calculates cancer prevalences
  prevalences[[i]] <- mean(raw_prevalences_stored)

  # saving raw point estimates and standard errors of prospective and retrospective
  new_pro_PE.RESULT[[i]] <- pro_pe_df
  new_pro_SE.RESULT[[i]] <- pro_se_df

  new_ret_PE.RESULT[[i]] <- ret_pe_df
  new_ret_SE.RESULT[[i]] <- ret_se_df

  # Find Standard Deviation
  new_sd_pro_df[[i]] <- apply(pro_pe_df, MARGIN=2, FUN=sd)
  new_sd_ret_df[[i]] <- apply(ret_pe_df, MARGIN=2, FUN=sd)

  # Mean of Point Estimates
  new_pro_PE_avg_df[[i]] <- colMeans(pro_pe_df)
  new_ret_PE_avg_df[[i]] <- colMeans(ret_pe_df)
```


https://dx.doi.org/10.21037/ace-2025-3

```
  # Mean of Standard Errors
  new_pro_SE_avg_df[[i]] <- colMeans(pro_se_df)
  new_ret_SE_avg_df[[i]] <- colMeans(ret_se_df)

  # ================ Coverage Rates

  # Prospective
  new_pro_b1_cov[[i]] <- find_coverage_rates(point_estimates_mtrx = pro_pe_df,
                                  std_err_mtrx = pro_se_df,
                                  betas_vec = betas,
                                  beta_index = 2,
                                  cnf_lvl = 1.96,
                                  sim_num = new_sim_num)

  new_pro_b2_cov[[i]] <- find_coverage_rates(point_estimates_mtrx = pro_pe_df,
                                  std_err_mtrx = pro_se_df,
                                  betas_vec = betas,
                                  beta_index = 3,
                                  cnf_lvl = 1.96,
                                  sim_num = new_sim_num)

  # Retrospective
  new_ret_b1_cov[[i]] <- find_coverage_rates(point_estimates_mtrx = ret_pe_df,
                                  std_err_mtrx = ret_se_df,
                                  betas_vec = betas,
                                  beta_index = 2,
                                  cnf_lvl = 1.96,
                                  sim_num = new_sim_num)

  new_ret_b2_cov[[i]] <- find_coverage_rates(point_estimates_mtrx = ret_pe_df,
                                  std_err_mtrx = ret_se_df,
                                  betas_vec = betas,
                                  beta_index = 3,
                                  cnf_lvl = 1.96,
                                  sim_num = new_sim_num)

  cat("\n Iteration:",i)
}
```


https://dx.doi.org/10.21037/ace-2025-3

```
#The following code runs the simulations for differential misclassification.
#Scenario with perfect Sp and differential Se:

large_pop <- basic_data_gen(row_num = 1e6, true_betas_vect = true_betas)

# CALCULATING S1 AND C1
sens_spec_settings <- calc_sens_spec_x1(population_simulated = large_pop,
                                        given_S0 = 0.9,
                                        given_C0 = 1.0,
                                        overall_sens = 0.8,
                                        overall_spec = 0.995)

# NOTE: Sensitivities
S0 = sens_spec_settings[1] # 0.9
S1 = sens_spec_settings[2] # 0.757

# NOTE: Specificities
C0 = sens_spec_settings[3] # 1.0
C1 = sens_spec_settings[4] # 0.9

spec_set = 1.0

# Variables and Lists
pop_size = 50000
sample_size = num_sims = 500
pro_pe_ethn_scen1_df = pro_se_ethn_scen1_df = ret_pe_ethn_scen1_df = ret_se_ethn_scen1_df <- blank_mtrx(n_sims =
num_sims, n_col = 3,
                                                                    c("Beta0", "Beta1", "Beta2"))

calcd_sens_spec_X1.0_scen1 = NULL
calcd_sens_spec_X1.1_scen1 = NULL
pops_saved_scen1 <- NULL

# ============== SCENARIO 1: IMPERFECT SENSITIVITY

for (y in 1:num_sims){
  set.seed(100+y)
  pop <- basic_data_gen(row_num=pop_size, true_betas_vect = true_betas)

  data_X1_equals0 <- pop[pop$X1==0,]
  data_X1_equals1 <- pop[pop$X1==1,]

  # S0 = 0.9
  data_X1_equals0$Ystar = generating_Ystar(Y_vect = data_X1_equals0$Y,
                                           sens_setting = S0,
                                           spec_setting = spec_set)
```


https://dx.doi.org/10.21037/ace-2025-3

```
  #S1 = 0.757
  data_X1_equals1$Ystar = generating_Ystar(Y_vect = data_X1_equals1$Y,
                                  sens_setting = S1,
                                  spec_setting = spec_set)

  calcd_sens_spec_X1.0_scen1[y] <- testfun(data_X1_equals0$Y,data_X1_equals0$Ystar)
  calcd_sens_spec_X1.1_scen1[y] <- testfun(data_X1_equals1$Y,data_X1_equals1$Ystar)

  cat("\n When X1=0, Se and Sp settings:",testfun(data_X1_equals0$Y,data_X1_equals0$Ystar))
  cat("\n When X1=1, Se and Sp settings:",testfun(data_X1_equals1$Y,data_X1_equals1$Ystar))

  complete_pop <- rbind(data_X1_equals0, data_X1_equals1)

  pops_saved_scen1[y] <- complete_pop

  sample <- sample_gen(population=complete_pop, samp_size = sample_size)

  # run logistic with prospective analysis
  prospec_coefs_scen1 <- new_cal_glm(pop_dataset = complete_pop, c = 4)

  pro_pe_ethn_scen1_df[y, ] <- prospec_coefs_scen1[,1]
  pro_se_ethn_scen1_df[y, ] <- prospec_coefs_scen1[,2]

  # run logistic with retrospective analysis
  retro_coefs_scen1 <- retro_logit(test_sample = sample)

  ret_pe_ethn_scen1_df[y, ] <- retro_coefs_scen1[,1]
  ret_se_ethn_scen1_df[y, ] <- retro_coefs_scen1[,2]

}


#================= COHORT

# Mean of Point Estimates
pro_PE_ethn_scen1_avg_df <- colMeans(pro_pe_ethn_scen1_df)
print(pro_PE_ethn_scen1_avg_df)

# Standard Deviation of Point Estimates
pro_pe_ethn_scen1_sd_df <- apply(pro_pe_ethn_scen1_df, MARGIN=2, FUN=sd)
print(pro_pe_ethn_scen1_sd_df)

# Mean of Standard Errors
pro_SE_ethn_scen1_avg_df <- colMeans(pro_se_ethn_scen1_df)
print(pro_SE_ethn_scen1_avg_df)

pro_ethn_scen1_B1_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_ethn_scen1_df,
                                  std_err_mtrx = pro_se_ethn_scen1_df,
```


https://dx.doi.org/10.21037/ace-2025-3

```
                                betas_vec = true_betas,
                                beta_index = 2,
                                cnf_lvl = 1.96,
                                sim_num = num_sims)

pro_ethn_scen1_B2_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_ethn_scen1_df,
                                std_err_mtrx = pro_se_ethn_scen1_df,
                                betas_vec = true_betas,
                                beta_index = 3,
                                cnf_lvl = 1.96,
                                sim_num = num_sims)
print(pro_ethn_scen1_B1_cov)
print(pro_ethn_scen1_B2_cov)

# ====================== CASE CONTROL

# Mean of Point Estimates
ret_PE_ethn_scen1_avg_df <- colMeans(ret_pe_ethn_scen1_df)
print(ret_PE_ethn_scen1_avg_df) # Beta 0 is far from -3.5

ret_pe_ethn_scen1_sd_df <- apply(ret_pe_ethn_scen1_df, MARGIN=2, FUN=sd)
print(ret_pe_ethn_scen1_sd_df)

# Mean of Standard Errors
ret_SE_ethn_scen1_avg_df <- colMeans(ret_se_ethn_scen1_df)
print(ret_SE_ethn_scen1_avg_df)

ret_ethn_scen1_B1_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_ethn_scen1_df,
                                std_err_mtrx = ret_se_ethn_scen1_df,
                                betas_vec = true_betas,
                                beta_index = 2,
                                cnf_lvl = 1.96,
                                sim_num = num_sims)

ret_ethn_scen1_B2_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_ethn_scen1_df,
                                std_err_mtrx = ret_se_ethn_scen1_df,
                                betas_vec = true_betas,
                                beta_index = 3,
                                cnf_lvl = 1.96,
                                sim_num = num_sims)

print(ret_ethn_scen1_B1_cov)
print(ret_ethn_scen1_B2_cov)



#Scenario with perfect Sp and differential Se where Se values for X_1=0 and X_1=1 were switched:
```


https://dx.doi.org/10.21037/ace-2025-3

```
# Switching Se Values for X1_1 and X1_0

row_num = pop_size = 50000
num_sims = 500
true_betas <- c(-3.5,1,1)
large_pop <- basic_data_gen(row_num = 1e6, true_betas_vect = true_betas)

sens_spec_settings <- calc_sens_spec_x1(population_simulated = large_pop,
                                        given_S0 = 0.9,
                                        given_C0 = 1.0,
                                        overall_sens = 0.8,
                                        overall_spec = 0.995)

S0 = sens_spec_settings[1] # 0.9
S1 = sens_spec_settings[2] # 0.757
C0 = sens_spec_settings[3] # 1.0
C1 = sens_spec_settings[4] # 0.99

spec_set = 1.0

pro_pe_switch_diff_sens_df = pro_se_switch_diff_sens_df = ret_pe_switch_diff_sens_df = ret_se_switch_diff_sens_df <-
blank_mtrx(n_sims = num_sims, n_col = 3,
                                                                    c("Beta0", "Beta1", "Beta2"))
samps_saved_switch_diff_sens = NULL
pops_saved_switch_diff_sens = NULL

### SIMULATIONS
# Perfect Specificity, Imperfect Differential Sensitivity

for (m in 1:num_sims){
  set.seed(m)
  pop <- basic_data_gen(row_num=pop_size, true_betas_vect = true_betas)

  data_X1.0_diff_sens <- pop[pop$X1==0,]
  data_X1.1_diff_sens <- pop[pop$X1==1,]

  # Mess-up X1 for Differential Specificity
  # switched sens=0.757 for when X1=0 and sens=0.9 when X1=1
  data_X1.0_diff_sens$Ystar <- generating_Ystar(Y_vect = data_X1.0_diff_sens$Y,
                                        sens_setting = S1,
                                        spec_setting = spec_set)

  data_X1.1_diff_sens$Ystar <- generating_Ystar(Y_vect = data_X1.1_diff_sens$Y,
                                        sens_setting = S0,
                                        spec_setting = spec_set)

  complete_pop <- rbind(data_X1.0_diff_sens, data_X1.1_diff_sens)
```


https://dx.doi.org/10.21037/ace-2025-3

```
  sample_found <- sample_gen(population=complete_pop, samp_size = 500)

  pops_saved_switch_diff_sens[[m]] <- complete_pop
  samps_saved_switch_diff_sens[[m]] <- sample_found

  # run logistic with prospective analysis
  # only produces 2 point estimates, not three
  analysis_diff_sens <- glm(Ystar ~ (X1 + X2), family="binomial", data = complete_pop)
  diff_sens_coefs <- summary(analysis_diff_sens)$coef

  pro_pe_switch_diff_sens_df[m,] <- diff_sens_coefs[,1]
  pro_se_switch_diff_sens_df[m,] <- diff_sens_coefs[,2]

  # run logistic with retrospective analysis
  samp_analysis_diff_sens <- glm(Ystarstar ~ (X1 + X2), family = "binomial", data = sample_found)
  samp_diff_sens_coefs <- summary(samp_analysis_diff_sens)$coef

  ret_pe_switch_diff_sens_df[m,] <- samp_diff_sens_coefs[,1]
  ret_se_switch_diff_sens_df[m,] <- samp_diff_sens_coefs[,2]

}
# Point Estimates
pro_PE_switch_diff_sens_avg <- colMeans(pro_pe_switch_diff_sens_df)
print(pro_PE_switch_diff_sens_avg)

ret_PE_switch_diff_sens_avg <- colMeans(ret_pe_switch_diff_sens_df)
print(ret_PE_switch_diff_sens_avg)

# Standard Deviation of Point Estimates
pro_pe_switch_diff_sens_sd <- apply(pro_pe_switch_diff_sens_df, MARGIN=2, FUN=sd)
print(pro_pe_switch_diff_sens_sd)

ret_pe_switch_diff_sens_sd <- apply(ret_pe_switch_diff_sens_df, MARGIN=2, FUN=sd)
print(ret_pe_switch_diff_sens_sd)

# Mean of Standard Errors
pro_SE_switch_diff_sens_avg <- colMeans(pro_se_switch_diff_sens_df)
print(pro_SE_switch_diff_sens_avg)

ret_SE_switch_diff_sens_avg <- colMeans(ret_se_switch_diff_sens_df)
print(ret_SE_switch_diff_sens_avg)

# COVERAGE RATES
pro_switch_diff_sens_B1_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_switch_diff_sens_df,
                                    std_err_mtrx = pro_se_switch_diff_sens_df,
                                    betas_vec = true_betas,
                                    beta_index = 2,
                                    cnf_lvl = 1.96,
```

 https://dx.doi.org/10.21037/ace-2025-3

```
                                    sim_num = num_sims)

pro_switch_diff_sens_B2_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_switch_diff_sens_df,
                                    std_err_mtrx = pro_se_switch_diff_sens_df,
                                    betas_vec = true_betas,
                                    beta_index = 3,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

print(pro_switch_diff_sens_B1_cov)
print(pro_switch_diff_sens_B2_cov)

ret_switch_diff_sens_B1_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_switch_diff_sens_df,
                                    std_err_mtrx = ret_se_switch_diff_sens_df,
                                    betas_vec = true_betas,
                                    beta_index = 2,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

ret_switch_diff_sens_B2_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_switch_diff_sens_df,
                                    std_err_mtrx = ret_se_switch_diff_sens_df,
                                    betas_vec = true_betas,
                                    beta_index = 3,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

print(ret_switch_diff_sens_B1_cov)
print(ret_switch_diff_sens_B2_cov)


# ============= SCENARIO 2: IMPERFECT SPECIFICITY

#Simulation code for perfect Se and differential Sp

true_betas <- c(-3.5, 1,1)

large_pop <- basic_data_gen(row_num = 1e6, true_betas_vect = true_betas)

sens_spec_settings <- calc_sens_spec_x1(population_simulated = large_pop,
                                given_S0 = 0.9,
                                given_C0 = 1.0,
                                overall_sens = 0.8,
                                overall_spec = 0.995)

S0 = sens_spec_settings[1] # 0.9
S1 = sens_spec_settings[2] # 0.757
C0 = sens_spec_settings[3] # 1.0
C1 = sens_spec_settings[4] # 0.99
```


https://dx.doi.org/10.21037/ace-2025-3

```
sens_set = 1.0

# Variables and Lists
pop_size = 50000
sample_size = num_sims = 500


calcd_sens_spec_X1.0_scen2 = NULL
calcd_sens_spec_X1.1_scen2 = NULL
pops_saved_scen2 <- NULL
samps_saved_scen2 <- NULL

pro_pe_ethn_scen2_df = pro_se_ethn_scen2_df = ret_pe_ethn_scen2_df = ret_se_ethn_scen2_df <- blank_mtrx(n_sims =
num_sims, n_col = 3,
                                                                    c("Beta0", "Beta1", "Beta2"))

### SIMULATIONS

for (s in 1:num_sims){
  set.seed(1000+s)
  pop <- basic_data_gen(row_num=pop_size, true_betas_vect = true_betas)

  data_X1_equals0 <- pop[pop$X1==0,]
  data_X1_equals1 <- pop[pop$X1==1,]

  data_X1_equals0$Ystar = generating_Ystar(Y_vect = data_X1_equals0$Y,
                                sens_setting = sens_set,
                                spec_setting = C0)

  data_X1_equals1$Ystar = generating_Ystar(Y_vect = data_X1_equals1$Y,
                                sens_setting = sens_set,
                                spec_setting = C1)

  calcd_sens_spec_X1.0_scen2[[s]] <- testfun(data_X1_equals0$Y,data_X1_equals0$Ystar)
  calcd_sens_spec_X1.1_scen2[[s]] <- testfun(data_X1_equals1$Y,data_X1_equals1$Ystar)

  cat("\n When X1=0, Se and Sp settings:",testfun(data_X1_equals0$Y,data_X1_equals0$Ystar))
  cat("\n When X1=1, Se and Sp settings:",testfun(data_X1_equals1$Y,data_X1_equals1$Ystar))

  complete_pop <- rbind(data_X1_equals0, data_X1_equals1)

  sample_found <- sample_gen(population=complete_pop, samp_size = sample_size)

  pops_saved_scen2[[s]] <- complete_pop
  samps_saved_scen2[[s]] <- sample_found

  # run logistic with prospective analysis
```


https://dx.doi.org/10.21037/ace-2025-3

```
  prospec_coefs_scen2 <- new_cal_glm(pop_dataset = complete_pop, c = 4)

  pro_pe_ethn_scen2_df[s, ] <- prospec_coefs_scen2[,1]
  pro_se_ethn_scen2_df[s, ] <- prospec_coefs_scen2[,2]

  # run logistic with retrospective analysis
  retro_coefs_scen2 <- retro_logit(test_sample = sample_found)

  ret_pe_ethn_scen2_df[s, ] <- retro_coefs_scen2[,1]
  ret_se_ethn_scen2_df[s, ] <- retro_coefs_scen2[,2]

}

# ====================== COHORT

# Mean of Point Estimates
pro_PE_ethn_scen2_avg_df <- colMeans(pro_pe_ethn_scen2_df)
print(pro_PE_ethn_scen2_avg_df)

pro_pe_ethn_scen2_sd_df <- apply(pro_pe_ethn_scen2_df, MARGIN=2, FUN=sd)
print(pro_pe_ethn_scen2_sd_df)

# Mean of Standard Errors
pro_SE_ethn_scen2_avg_df <- colMeans(pro_se_ethn_scen2_df)
print(pro_SE_ethn_scen2_avg_df)

pro_ethn_scen2_B1_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_ethn_scen2_df,
                                  std_err_mtrx = pro_se_ethn_scen2_df,
                                  betas_vec = true_betas,
                                  beta_index = 2,
                                  cnf_lvl = 1.96,
                                  sim_num = num_sims)

pro_ethn_scen2_B2_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_ethn_scen2_df,
                                  std_err_mtrx = pro_se_ethn_scen1_df,
                                  betas_vec = true_betas,
                                  beta_index = 3,
                                  cnf_lvl = 1.96,
                                  sim_num = num_sims)

print(pro_ethn_scen2_B1_cov)
print(pro_ethn_scen2_B2_cov)

# ====================== CASE CONTROL

# Mean of Point Estimates
ret_PE_ethn_scen2_avg_df <- colMeans(ret_pe_ethn_scen2_df)
print(ret_PE_ethn_scen2_avg_df)
```


https://dx.doi.org/10.21037/ace-2025-3

```
ret_pe_ethn_scen2_sd_df <- apply(ret_pe_ethn_scen2_df, MARGIN=2, FUN=sd)
print(ret_pe_ethn_scen2_sd_df)

# Mean of Standard Errors
ret_SE_ethn_scen2_avg_df <- colMeans(ret_se_ethn_scen2_df)
print(ret_SE_ethn_scen2_avg_df)

ret_ethn_scen2_B1_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_ethn_scen2_df,
                                  std_err_mtrx = ret_se_ethn_scen2_df,
                                  betas_vec = true_betas,
                                  beta_index = 2,
                                  cnf_lvl = 1.96,
                                  sim_num = num_sims)

ret_ethn_scen2_B2_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_ethn_scen2_df,
                                  std_err_mtrx = ret_se_ethn_scen2_df,
                                  betas_vec = true_betas,
                                  beta_index = 3,
                                  cnf_lvl = 1.96,
                                  sim_num = num_sims)

print(ret_ethn_scen2_B1_cov)
print(ret_ethn_scen2_B2_cov)


#Scenario with perfect Se and differential Sp where Sp values for X_1=0 and X_1=1 were switched:

true_betas <- c(-3.5, 1,1)
large_pop <- basic_data_gen(row_num = 1e6, true_betas_vect = true_betas)

sens_spec_settings <- calc_sens_spec_x1(population_simulated = large_pop,
                                given_S0 = 0.9,
                                given_C0 = 1.0,
                                overall_sens = 0.8,
                                overall_spec = 0.995)

S0 = sens_spec_settings[1] # 0.9
S1 = sens_spec_settings[2] # 0.757
C0 = sens_spec_settings[3] # 1.0
C1 = sens_spec_settings[4] # 0.99

Sens_set <- 1.0

pro_pe_switch_diff_spec_df = pro_se_switch_diff_spec_df = ret_pe_switch_diff_spec_df = ret_se_switch_diff_spec_df <-blank_
mtrx(n_sims = num_sims, n_col = 3,
                                                                    c("Beta0", "Beta1", "Beta2"))
samps_saved_switch_diff_spec = NULL
```


https://dx.doi.org/10.21037/ace-2025-3

```
pops_saved_switch_diff_spec = NULL

### SIMULATIONS

for (w in 1:num_sims){
 set.seed(w)
 pop <- basic_data_gen(row_num=pop_size, true_betas_vect = true_betas)

 data_X1.0_diff_spec <- pop[pop$X1==0,]
 data_X1.1_diff_spec <- pop[pop$X1==1,]

 # Mess-up X1 for Differential Specificity
 # switched spec=1.0 for when X1=1 and spec=0.99 when X1=0
 data_X1.0_diff_spec$Ystar <- generating_Ystar(Y_vect = data_X1.0_diff_spec$Y,
                                  sens_setting = sens_set,
                                  spec_setting = C1)

 data_X1.1_diff_spec$Ystar <- generating_Ystar(Y_vect = data_X1.1_diff_spec$Y,
                                  sens_setting = sens_set,
                                  spec_setting = C0)

 complete_pop <- rbind(data_X1.0_diff_spec, data_X1.1_diff_spec)

 sample_found <- sample_gen(population=complete_pop, samp_size = 500)

 pops_saved_switch_diff_spec[[w]] <- complete_pop
 samps_saved_switch_diff_spec[[w]] <- sample_found

 # run logistic with prospective analysis
 # only produces 2 point estimates, not three
 analysis_diff_spec <- glm(Ystar ~ (X1 + X2), family="binomial", data = complete_pop)
 diff_spec_coefs <- summary(analysis_diff_spec)$coef

 pro_pe_switch_diff_spec_df[w,] <- diff_spec_coefs[,1]
 pro_se_switch_diff_spec_df[w,] <- diff_spec_coefs[,2]

 # run logistic with retrospective analysis
 samp_analysis_diff_spec <- glm(Ystarstar ~ (X1 + X2), family = "binomial", data = sample_found)
 samp_diff_spec_coefs <- summary(samp_analysis_diff_spec)$coef

 ret_pe_switch_diff_spec_df[w,] <- samp_diff_spec_coefs[,1]
 ret_se_switch_diff_spec_df[w,] <- samp_diff_spec_coefs[,2]

}

# Point Estimates
pro_PE_switch_diff_spec_avg <- colMeans(pro_pe_switch_diff_spec_df)
print(pro_PE_switch_diff_spec_avg)
```

 https://dx.doi.org/10.21037/ace-2025-3

```
ret_PE_switch_diff_spec_avg <- colMeans(ret_pe_switch_diff_spec_df)
print(ret_PE_switch_diff_spec_avg)

# Standard Deviation of Point Estimates
pro_pe_switch_diff_spec_sd <- apply(pro_pe_switch_diff_spec_df, MARGIN=2, FUN=sd)
print(pro_pe_switch_diff_spec_sd)

ret_pe_switch_diff_spec_sd <- apply(ret_pe_switch_diff_spec_df, MARGIN=2, FUN=sd)
print(ret_pe_switch_diff_spec_sd)

# Mean of Standard Errors
pro_SE_switch_diff_spec_avg <- colMeans(pro_se_switch_diff_spec_df)
print(pro_SE_switch_diff_spec_avg)

ret_SE_switch_diff_spec_avg <- colMeans(ret_se_switch_diff_spec_df)
print(ret_SE_switch_diff_spec_avg)

# COVERAGE RATES
pro_switch_diff_spec_B1_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_switch_diff_spec_df,
                                    std_err_mtrx = pro_se_switch_diff_spec_df,
                                    betas_vec = true_betas,
                                    beta_index = 2,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

pro_switch_diff_spec_B2_cov <- find_coverage_rates(point_estimates_mtrx = pro_pe_switch_diff_spec_df,
                                    std_err_mtrx = pro_se_switch_diff_spec_df,
                                    betas_vec = true_betas,
                                    beta_index = 3,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

print(pro_switch_diff_spec_B1_cov)
print(pro_switch_diff_spec_B2_cov)

ret_switch_diff_spec_B1_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_switch_diff_spec_df,
                                    std_err_mtrx = ret_se_switch_diff_spec_df,
                                    betas_vec = true_betas,
                                    beta_index = 2,
                                    cnf_lvl = 1.96,
                                    sim_num = num_sims)

ret_switch_diff_spec_B2_cov <- find_coverage_rates(point_estimates_mtrx = ret_pe_switch_diff_spec_df,
                                    std_err_mtrx = ret_se_switch_diff_spec_df,
                                    betas_vec = true_betas,
                                    beta_index = 3,
                                    cnf_lvl = 1.96,
```


https://dx.doi.org/10.21037/ace-2025-3

```
                                sim_num = num_sims)

print(ret_switch_diff_spec_B1_cov)
print(ret_switch_diff_spec_B2_cov)
```


https://dx.doi.org/10.21037/ace-2025-3