Inverse probability of censoring weighting (IPCW) for linear regression

Brice Ozenne

October 8, 2021

1 Principle

Inverse probability of censoring weighting (IPCW) is a method able to handle informative drop-out. Intuitively, in presence of informative drop-out a complete case analysis is a biased approach as individuals with complete data are not representative of the population. However with an appropriate re-weighting of the individuals with complete data, we can "re-balance" our sample and make it representative of the population. To do so, we divide the population into sub-populations and attribute weights to individuals who did not drop-out inversely proportional to the frequency of the drop-out in the sub-population. Thanks to the weights, individuals who did not drop-out "represent" the individuals who dropped-out. Thus, overall, the weighted sample is representative of the population.

In this document, we will illustrate the use of IPCW:

- with a continuous outcome and compare it with a full information approach (via a mixed model).
- with a binary outcome.

2 Continuous outcome

2.1 Generative model

To illustrate the use of IPCW in the continuous outcome case, we will consider a longitudinal study with 2 groups ($G = 0$ and $G = 1$) and 2 timepoints (t_1 and t_2) and no other covariate. The outcome Y is normally distributed, denoted Y_1 at t_1 and Y_2 at t_2:

$$\begin{bmatrix} Y_1|G=0 \\ Y_2|G=0 \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} 50 \\ 50 - d\mu_1 \end{bmatrix}, 100\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} Y_1|G=1 \\ Y_2|G=1 \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} 75 \\ 75 - d\mu_2 \end{bmatrix}, 100\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right)$$

At time t_2 we may not observed Y due to drop-out. The probability of drop-out depends either:

- on the (latent) group: π_{C_1} in $G = 0$ and π_{C_2} in $G = 1$
- on the baseline value: $\frac{1}{1+\exp(-\pi_{C_1}(Y_1-62.5)/10}$ in $G = 0$
  $\frac{1}{1+\exp(-\pi_{C_2}(Y_1-62.5)/10)}$ in $G = 1$

The corresponding R function is given in the next page. It uses the following arguments:

- `n`: [positive integer] number of patient per group
- `rho`: [numeric between -1 and 1] correlation over time
- `dmu`: [numeric vector of length 2] change in mean in each group
- `causeC`: [character] variable defining the censoring probability (either `latent` or `baseline`)
- `piC`: [numeric vector of length 2, between 0 and 1] probability of drop-out in each group (`latent`) or regression parameter for this probability (`baseline`)

```r
simTrial <- function(n, rho, dmu, causeC, piC){
  ## load packages and check user input
  require(mvtnorm)
  require(data.table)
  causeC <- match.arg(causeC, c("baseline","latent"))
  ## simulate data
  sigma <- 10
  Sigma <- sigma^2*matrix(c(1,rho,rho,1),2,2)
  ## gather into dataset
  M.Ym <- rmvnorm(n, mean = c(50, 50-dmu[1]), sigma = Sigma)
  M.Ys <- rmvnorm(n, mean = c(75, 75-dmu[2]), sigma = Sigma)
  dtL <- rbind(
    data.table(id = 1:n, mdd = "moderate", time = "T1", Y = M.Ym[,1]),
    data.table(id = 1:n, mdd = "moderate", time = "T2", Y = M.Ym[,2]),
    data.table(id = n+(1:n), mdd = "severe", time = "T1", Y = M.Ys[,1]),
    data.table(id = n+(1:n), mdd = "severe", time = "T2", Y = M.Ys[,2])
  )
  ## define probability of dropout
  dtL$probaDO <- 0
  if(causeC == "latent"){
    dtL[time=="T2", probaDO := ifelse(.SD$mdd=="moderate",piC[1],piC[2])]
  }else if(causeC == "baseline"){
    dtL$res <- 0.1 ## no used - just  to initialize
    dtL[mdd=="moderate", res := c((Y[1]-62.5)/sigma,NA), by = "id"]
    dtL[mdd=="severe", res := c((Y[1]-62.5)/sigma,NA), by = "id"]
    dtL[mdd=="moderate", probaDO := c(0,plogis(piC[1]*res[1])), by = "id"]
    dtL[mdd=="severe", probaDO := c(0,plogis(piC[2]*res[1])), by = "id"]
    dtL$res <- NULL
  }
  ## simulate dropout
  dtL[,dropout := rbinom(.N,prob=probaDO,size=1)]
  dtL[,Yobs:=Y]
  dtL[dropout==1,Yobs:=NA]
  ## export
  dtL$probaDO <- NULL
  setkeyv(dtL,"id")
  return(dtL)
}
```

2.2 Illustrative example 1

Consider a study were we follow depressed individual over time. They have a baseline measurement, then are given a treatment, and then have a follow-up measurement. We would like to assess the treatment effect in term of depression score [1]. The population of interest contain severely and moderately depressed individuals; the treatment may work differently in each sub-population. Unfortunately, some study participants dropped-out and it seems that they are more likely to drop-out when their baseline score is high.

We can simulate such a dataset using the following function:

```
set.seed(10)
dtL.B <- simTrial(n = 1000, rho = 0.8, dmu = c(25,50),
    causeC = "baseline", piC = c(1,1))
head(dtL.B)
```

```
   id      mdd time        Y dropout     Yobs
1:  1 moderate   T1 49.34367       0 49.34367
2:  1 moderate   T2 23.43583       0 23.43583
3:  2 moderate   T1 35.05489       0 35.05489
4:  2 moderate   T2 13.50810       0 13.50810
5:  3 moderate   T1 54.37770       0 54.37770
6:  3 moderate   T2 29.80367       1       NA
```

Here we have simulated a two sub-populations of 1000, with a correlation of 0.8 between baseline and follow-up. The treatment effect is twice bigger for the severely depressed population but individuals from this population are also much more likely to drop-out as they tend to have higher baseline score. So we expect complete case estimators to be downward biased.

Without drop-out, we could use a simple linear model to carry-out the analysis:

```
dtW.Boracle <- dcast(dtL.B, formula = id ~ time, value.var = "Y")
dtW.Boracle$diff <- dtW.Boracle$T2-dtW.Boracle$T1
e.Boracle <- lm(diff~1, data = dtW.Boracle)
summary(e.Boracle)$coef
```

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -37.34478  0.3131657 -119.2492        0
```

[1]:To simplify, there is no control group - we assume that without treatment the depression score would be constant.

leading to an estimate quite close to the true value:

```
(-25-50)/2
```

```
[1] -37.5
```

With drop-out, a complete case analysis would lead to a biased estimator. In this example, we can "see" that the estimated value is far away from the true one (even when accouting for the uncertainty):

```
dtW.B <- dcast(dtL.B, formula = id + mdd ~ time, value.var = "Yobs")
dtW.B$diff <- dtW.B$T2-dtW.B$T1
dtW.BCC <- dtW.B[!is.na(diff)]
e.BCC <- lm(diff~1, data = dtW.BCC)
summary(e.BCC)$coef
```

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -30.98307  0.3889309 -79.66214        0
```

An alternative approach would be to use a linear mixed model (i.e. full information):

```
require(nlme)
e.BFI <- lme(Yobs~time, random = ~1|id, data = dtL.B,
      na.action = na.omit)
summary(e.BFI)$tTable
```

```
                Value Std.Error   DF   t-value p-value
(Intercept)  62.39901 0.3268707 1999 190.89814       0
timeT2      -34.72137 0.3922177 1030 -88.52576       0
```

which appears better than the complete case analysis but still downward biased. This can be a bit surprising at first, but can be explained when seeing the mixed model as a way to "impute" missing values at follow-up. The current mixed model is misspecified (missing interaction between time and group) and it therefore use the wrong imputation model. This is illustrated in Figure 1 (see appendix A for the R code).

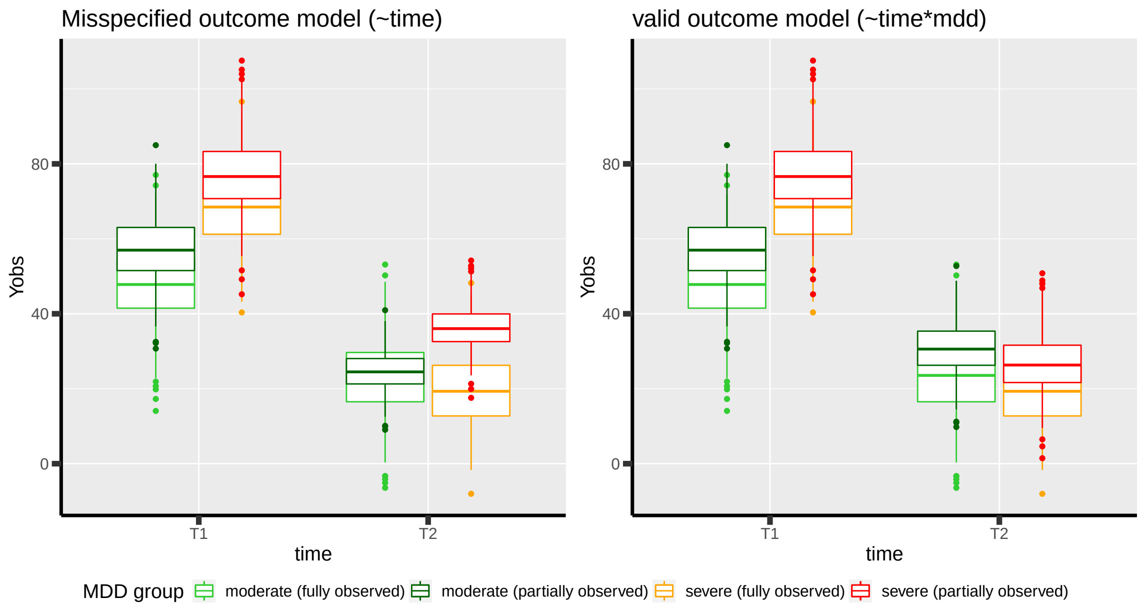

Figure 1: Distribution of the observed and imputed value when using the mixed model.

With a correct model for the outcome (i.e. adding the interaction), the mixed would be able to inpute the observations in an unbiased way:

```
e.BFIoracle <- lme(Yobs~time*mdd, random = ~1|id, data = dtL.B,
     na.action = na.omit)
summary(e.BFIoracle)$tTable
```

```
                      Value Std.Error   DF    t-value       p-value
(Intercept)        50.14399 0.3201715 1998  156.61602  0.000000e+00
timeT2            -24.97957 0.2351254 1029 -106.23938  0.000000e+00
mddsevere          24.51004 0.4527909 1998   54.13102  0.000000e+00
timeT2:mddsevere -24.90905 0.4443197 1029  -56.06111 4.849765e-315
```

which would lead to a much better estimator:

```
library(multcomp)
glht(e.BFIoracle, linfct = "timeT2+0.5*timeT2:mddsevere=0")
```

```
         General Linear Hypotheses

Linear Hypotheses:
                                   Estimate
timeT2 + 0.5 * timeT2:mddsevere == 0   -37.43
```

An alternative approach that does not require to specify an outcome model is to use IPCW. It instead requires to correctly specify a model for the probability of not dropping out at follow-up:

```
dtW.B$observed <- !is.na(dtW.B$T2)
e.glmW.B <- glm(observed ~ T1, data = dtW.B,
  family = binomial(link = "logit"))
coef(e.glmW.B)
```

```
(Intercept)          T1
  6.6357425  -0.1047988
```

and then compute the weights for observations with full data:

```
dtW.B$weight.oracle <- 1/predict(e.glmW.B, newdata = dtW.B,
     type = "response")
dtW.B[observed == TRUE, sum(weight.oracle)]
```

```
[1] 2045.06
```

Note that the weights almost sum to the total sample size. We then perform the complete case analysis with these weights:

```
dtW.BCC <- dtW.B[!is.na(diff)]
e.BIPCW <- lm(diff~1, data = dtW.BCC, weights = dtW.BCC$weight.oracle)
summary(e.BIPCW)$coef
```

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -37.84241  0.4369635 -86.60314        0
```

which gives a result very close to the true value. Here the IPCW works very well because we have specified the correct censoring model.

2.3 Illustrative example 2

Consider a similar study with a different cause of drop-out. This time drop-out is not due to baseline value but due to the severity of the disease (i.e. group): two patients severely depressed but with different baseline score will have exactly the same probability of drop-out while two patients, one severely depressed and the other moderately depressed, with same baseline score will have different probability of drop-out.

We can simulate such a dataset using the following function:

```
set.seed(10)
dtL.L <- simTrial(n = 1000, rho = 0.8, dmu = c(25,50),
    causeC = "latent", piC = c(0.2,0.7))
print(dtL.L)
```

```
        id      mdd time         Y dropout     Yobs
   1:    1 moderate   T1 49.34367       0 49.34367
   2:    1 moderate   T2 23.43583       0 23.43583
   3:    2 moderate   T1 35.05489       0 35.05489
   4:    2 moderate   T2 13.50810       0 13.50810
   5:    3 moderate   T1 54.37770       0 54.37770
  ---
3996: 1998   severe   T2 26.26605       1       NA
3997: 1999   severe   T1 70.81751       0 70.81751
3998: 1999   severe   T2 15.46369       1       NA
3999: 2000   severe   T1 73.53750       0 73.53750
4000: 2000   severe   T2 23.75026       1       NA
```

Overall the expected treatment effect is the same as before and, without drop-out, the linear model gives the same estimates:

```
dtW.Loracle <- dcast(dtL.L, formula = id ~ time, value.var = "Y")
dtW.Loracle$diff <- dtW.Loracle$T2-dtW.Loracle$T1
e.Loracle <- lm(diff~1, data = dtW.Loracle)
summary(e.Loracle)$coef
```

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -37.34478  0.3131657 -119.2492        0
```

With drop-out, a complete case analysis would still lead to a downward biased estimator:

```
dtW.L <- dcast(dtL.L, formula = id + mdd ~ time, value.var = "Yobs")
dtW.L$diff <- dtW.L$T2-dtW.L$T1
dtW.LCC <- dtW.L[!is.na(diff)]
e.LCC <- lm(diff~1, data = dtW.LCC)
summary(e.LCC)$coef
```

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -31.47144  0.3853402 -81.67182        0
```

for a reason similar as before, as patients from the severely depressed group will drop more often and they benefit more from the treatmet. We can use a linear mixed model (i.e. full information):

```
require(nlme)
e.LFI <- lme(Yobs~time, random = ~1|id, data = dtL.L, na.action = na.omit
   )
summary(e.LFI)$tTable
```

```
                Value Std.Error   DF   t-value p-value
(Intercept)  62.39901 0.3216035 1999 194.02463       0
timeT2      -33.90248 0.3789931 1090 -89.45409       0
```

which is better than the complete case analysis still biased because once more the outcome model is misspecified. With a correctly specified outcome model, we would get a much better estimate:

```
e.LFIoracle <- lme(Yobs~time*mdd, random = ~1|id, data = dtL.L, na.action
    = na.omit)
glht(e.LFIoracle, linfct = "timeT2+0.5*timeT2:mddsevere=0")
```

```

	 General Linear Hypotheses

Linear Hypotheses:
                                  Estimate
timeT2 + 0.5 * timeT2:mddsevere == 0    -37.3
```

When using IPCW, we should model the probability of not dropping out at follow-up as a function of the latent group:

```
dtW.L$observed <- !is.na(dtW.L$T2)
e.glmW.Loracle <- glm(observed ~ mdd, data = dtW.L,
       family = binomial(link = "logit"))
```

and then compute the weights for observations with full data:

```
dtW.L$weight.oracle <- 1/predict(e.glmW.Loracle, newdata = dtW.L,type = "
   response")
dtW.L[observed == TRUE, sum(weight.oracle)]
```

```
[1] 2000
```

Note that the weights sum to the total sample size. We then perform the complete case analysis with these weights:

```
dtW.LCC <- dtW.L[!is.na(diff)]
e.LIPCWoracle <- lm(diff~1, data = dtW.LCC, weights = dtW.LCC$weight.
   oracle)
summary(e.LIPCWoracle)$coef
```

```
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) -37.35191  0.4242621 -88.0397        0
```

which gives a result very close to the true value. A more feasible IPCW would use the baseline score to define the weights:

```
e.glmW.L <- glm(observed ~ T1, data = dtW.L,
       family = binomial(link = "logit"))
dtW.L$weight <- 1/predict(e.glmW.L, newdata = dtW.L, type = "response")
dtW.L[observed == TRUE, sum(weight)]
```

```
[1] 2038.825
```

We then perform the complete case analysis with these new weights:

```
dtW.LCC <- dtW.L[!is.na(diff)]
e.LIPCW <- lm(diff~1, data = dtW.LCC, weights = dtW.LCC$weight)
summary(e.LIPCW)$coef
```

```
            Estimate Std. Error   t value Pr(>|t|)
(Intercept) -36.0517  0.4258783 -84.65258        0
```

2.4 Simulation study

The quality of the previous estimators is compared using a simulation study. The results are summarized by Figure 2.

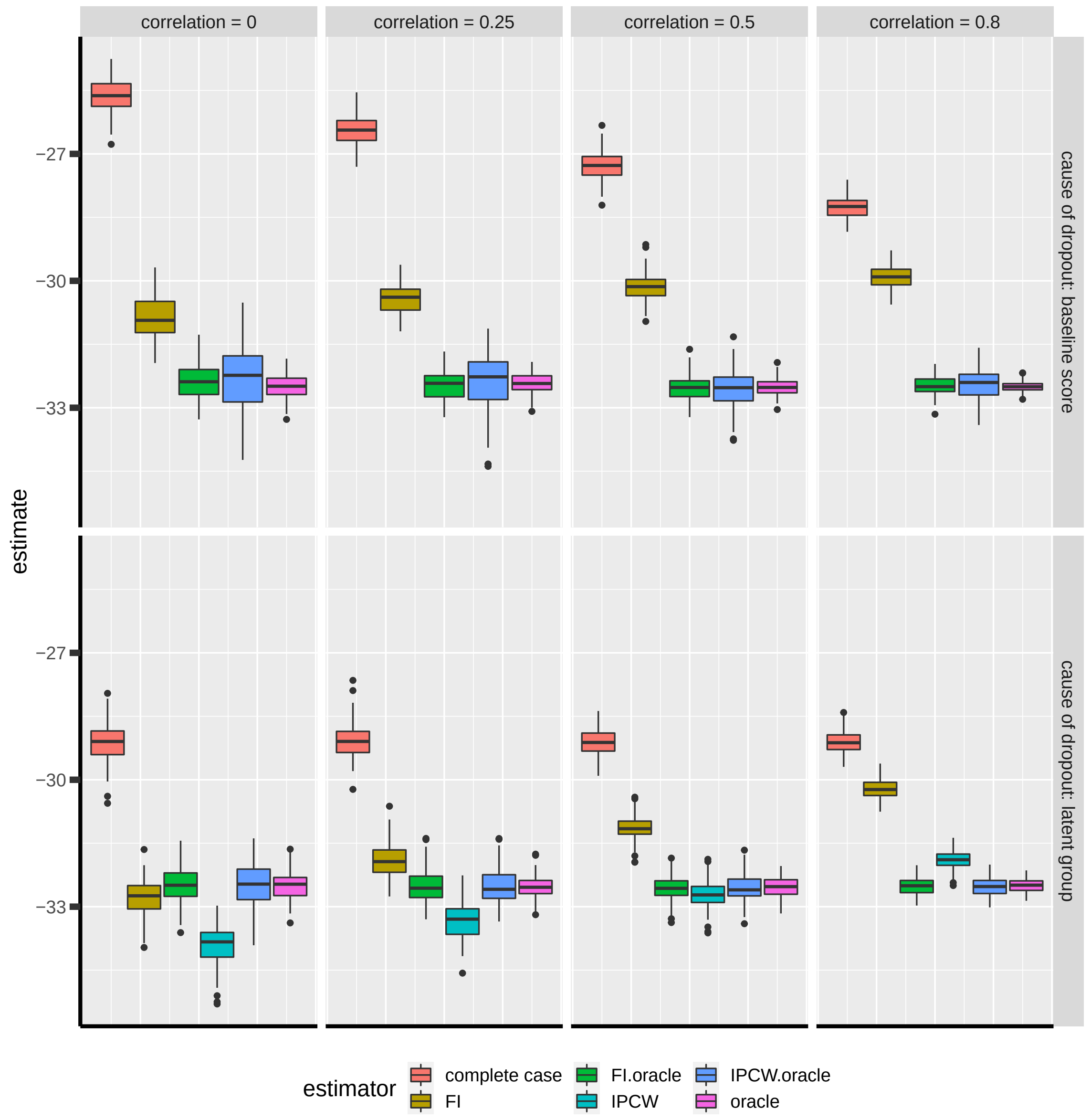

Figure 2: Comparison between the empirical distributions of the estimators (Gaussian case) for a sample size of 1000 using 100 datasets. FI: full information (random intercept model), IPCW: inverse probability of censoring weights.

3 Binary outcome

3.1 Illustrative example

A somehow similar approach can be used for binary endpoints. Consider now a study comparing the survival probability at 1 year of patients treated with a new drug vs. standard care. The population is composed of two types of patients, say some with hypertension and some without. Survival as well as the treatment effect may differ depending of the hypertension status. Hypertension may also affect the drop-out probability.

We can simulate such a dataset using the following function:

```
simTrial <- function(n, dmu, dpC){
  require(BuyseTest)
  require(data.table)
  ## simulate data
  dt1  <- simBuyseTest(n.T = n, n.C = n,
         argsBin = NULL, argsCont = NULL,
         argsTTE = list(scale.T = 1+dmu[1],
          scale.C = 1,
          scale.Censoring.T = 1+dpC[1],
          scale.Censoring.C = 1),
         latent = TRUE)
  dt2  <- simBuyseTest(n.T = n, n.C = n,
         argsBin = NULL, argsCont = NULL,
         argsTTE = list(scale.T = 2+dmu[2],
          scale.C = 2,
          scale.Censoring.T = 2+dpC[2],
          scale.Censoring.C = 2),
         latent = TRUE)
  ## gather into dataset
  dt <- rbind(
    cbind(id = 1:NROW(dt1), group = "G1", dt1),
    cbind(id = NROW(dt1) + 1:NROW(dt2), group = "G2", dt2)
  )
  return(dt)
}
set.seed(11)
tau <- 1

dt <- simTrial(n = 1000, dmu = c(0,1), dpC = c(0,1))
dt$responseUncensored <- dt$eventtimeUncensored<=tau
dt$response <- ifelse((dt$status==1)+(dt$eventtime>tau),dt$eventtime<=tau,
   NA)
dt$observed <- ifelse((dt$status==1)+(dt$eventtime>tau),1,0)
print(dt)
```

```
        id group treatment eventtimeUncensored eventtimeCensoring  eventtime
   1:    1    G1         C          0.07747187          0.4441963 0.07747187
   2:    2    G1         C          0.18271259          0.3567996 0.18271259
   3:    3    G1         C          0.14864417          0.2298933 0.14864417
   4:    4    G1         C          0.26922419          0.6492349 0.26922419
   5:    5    G1         C          0.52950600          0.2238334 0.22383343
  ---
3996: 3996    G2         T          1.09150744          5.6892558 1.09150744
```

```
3997: 3997    G2         T          5.83550031         1.7693238 1.76932381
3998: 3998    G2         T          0.88964585         0.2485173 0.24851729
3999: 3999    G2         T          0.44492756         4.8949421 0.44492756
4000: 4000    G2         T         18.10666952         2.5876528 2.58765282
      status responseUncensored response observed
   1:      1               TRUE     TRUE        1
   2:      1               TRUE     TRUE        1
   3:      1               TRUE     TRUE        1
   4:      1               TRUE     TRUE        1
   5:      0               TRUE       NA        0
  ---
3996:      1              FALSE    FALSE        1
3997:      0              FALSE    FALSE        1
3998:      0               TRUE       NA        0
3999:      1               TRUE     TRUE        1
4000:      0              FALSE    FALSE        1
```

In absence of drop-out, we can compare the survival probabilities at 1 year using a logistic regression:

```
e.oracle <- glm(responseUncensored ~ treatment,
  data = dt, family = binomial(link="logit"))
summary(e.oracle)$coef
```

```
               Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  0.08204599 0.04475899  1.833062 6.679338e-02
treatmentT  -0.27060267 0.06341278 -4.267321 1.978345e-05
```

In presence of (differential) drop-out, a complete case analysis (i.e. restricting the analysis to the patients where the survival status at 1 year is known) would be biased:

```
dt.cc <- dt[dt$observed==1]
e.cc <- glm(response ~ treatment,
     data = dt.cc, family = binomial(link="logit"))
summary(e.cc)$coef
```

```
              Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  0.4008704 0.05727500  6.999047 2.577101e-12
treatmentT  -0.4222127 0.07955849 -5.306947 1.114767e-07
```

A first idea would be to re-use the IPCW approach, first fitting a logistic model for the probability of being observed at 1-year and then computing the weights:

```
e.IPCmodel <- glm(observed ~ group*treatment, data = dt, family = binomial
    (link="logit"))
dt$IPCweights <- 1/predict(e.IPCmodel, newdata = dt, type = "response")
sum(dt$IPCweights)
```

```
[1] 6305.334
```

The subsequent estimator will not be correct:

```
dt.cc <- dt[dt$observed==1]
e.IPCWcc <- glm(response ~ treatment, data = dt.cc,
  family = binomial(link="logit"), weights = dt.cc$IPCweights)
summary(e.IPCWcc)$coef
```

```
Advarselsbesked:
I eval(family$initialize) : non-integer #successes in a binomial glm!
              Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  0.4515849 0.04586621  9.845700 7.153939e-23
treatmentT  -0.3341242 0.06411408 -5.211402 1.874189e-07
```

as we disregarded the duration of observation among the censored individuals. Intuitively, individuals censored early are more at risk of dying and therefore should "transfer" more weight than those censored late, e.g. just before 1 year, who don't really need to transfer weights. This can be perform using a survival model (here a Cox model) and using as weights the inverse of the probability of not being censored at the earliest between when the event occured and 1 year:

```
library(survival)
library(riskRegression)
e.IPCmodel2 <- coxph(Surv(eventtime,status==0) ~ group*treatment,
       data = dt, x = TRUE, y = TRUE)
iPred <- predictCox(e.IPCmodel2, newdata = dt,
      time = pmin(dt$eventtime,tau)-(1e-12), diag = TRUE)$survival
dt$IPCweights2 <- dt$observed/iPred
sum(dt$IPCweights2)
```

```
[1] 3997.757
```

We can then use the weights in a logistic model:

```
dt.cc <- dt[dt$observed==1]
e.IPCWcc <- glm(response ~ treatment, data = dt.cc,
  family = quasibinomial(link="logit"), weights = dt.cc$IPCweights2)
summary(e.IPCWcc)$coef
```

```
               Estimate Std. Error    t value     Pr(>|t|)
(Intercept)  0.04110777 0.05561028  0.7392117 0.4598457572
treatmentT  -0.26472454 0.07902160 -3.3500276 0.0008196644
```

which is very close to the true value. Note that this estimator is implemented in the riskRegression package:

```
e.wglm <- wglm(regressor.event = ~treatment,
        formula.censor = Surv(eventtime,status==0)~group*treatment,
        times = 1,
        data = dt[,.(eventtime,status,group,treatment)])
summary(e.wglm)
```

```
     IPCW logistic regression :
----------------------------------------------------------------------------------
  - time: 1
glm(XX_status.1_XX ~ treatment, family = binomial(link = "logit"),
    weights = "XX_IPCW.1_XX")

               Estimate Std. Error   t value    Pr(>|t|)
(Intercept)  0.04110777 0.06102480  0.673624 0.500550414
treatmentT  -0.26472454 0.08416678 -3.145238 0.001659519
----------------------------------------------------------------------------------
```

This estimator is also implemented in the `mets` package[2]:

```
library(mets)
e.mets <- logitIPCW(formula = Event(eventtime,status) ~ treatment,
      cens.model = ~group*treatment,
      time = 1, data = dt, cens.code = 0, cause = 1)
e.mets
```

```

    n events
 4000   1409

 4000 clusters
coeffients:
             Estimate   Std.Err      2.5%     97.5% P-value
(Intercept)  0.041108  0.056878 -0.070371  0.152587  0.4698
treatmentT  -0.264725  0.082562 -0.426543 -0.102906  0.0013

exp(coeffients):
            Estimate    2.5%  97.5%
(Intercept)  1.04196 0.93205 1.1648
treatmentT   0.76742 0.65276 0.9022
```

[2]the standard errors are slightly different though

3.2 Simulation study

The quality of the previous estimators is compared using a simulation study. The results are summarized by Figure 3 and Figure 4

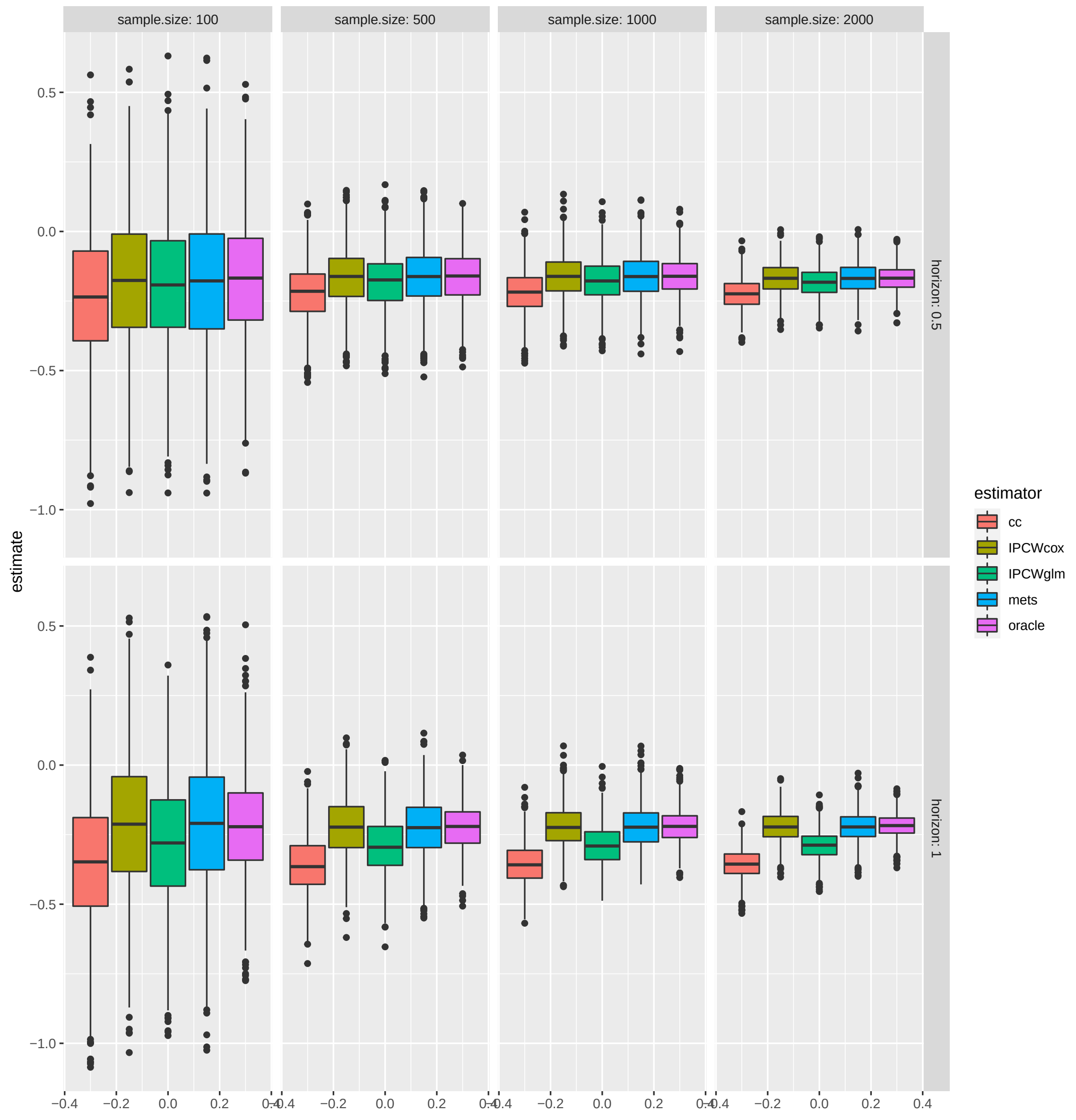

Figure 3: Comparison between the empirical distributions of the estimators (binary case) across sample size. Based on 1000 replicates.

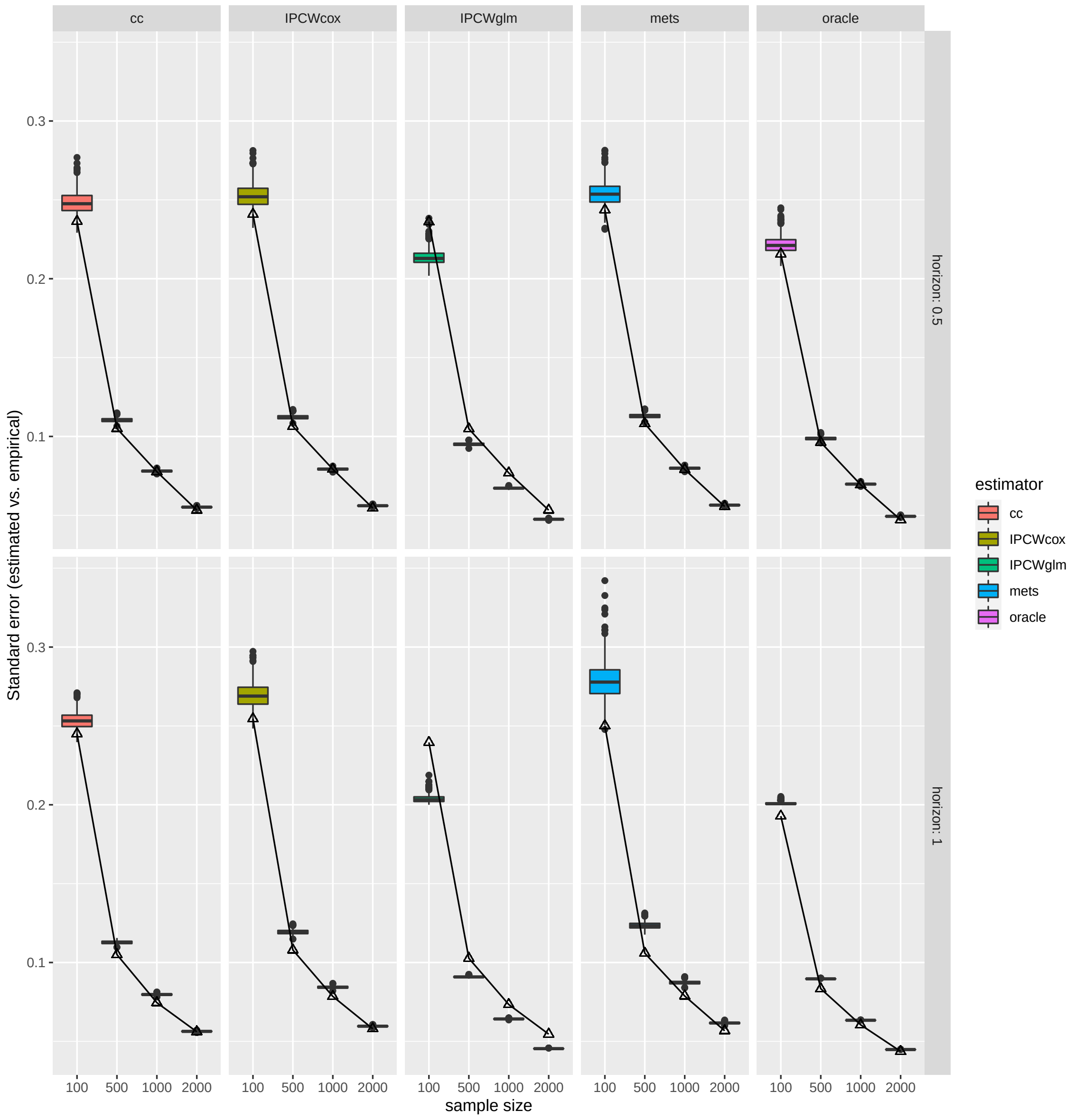

Figure 4: Comparison between the modeled standard errors of the estimates (boxplot) and the empirical ones (triangles linked by a line) across sample size. Based on 1000 replicates.

Appendix A: Graphical display of the imputation (Figure 1)

Alternative R code to fit a random intercept model

```
library(LMMstar)
e.lmm <- lmm(Yobs~time, repetition = ~time|id, structure = "CS",
      data = dtL.B)
eOracle.lmm <- lmm(Yobs~mdd*time, repetition = ~time|id, structure = "CS"
    ,
     data = dtL.B)
```

Identify patient with missing data:

```
dtL.BNA <- dtL.B[id %in% id.NA==TRUE]
dtL.BNA$group2 <- paste0(dtL.BNA$mdd," (partially observed)")
dtL.BNNA <- dtL.B[id %in% id.NA==FALSE]
dtL.BNNA$group2 <- paste0(dtL.BNNA$mdd," (fully observed)")
```

Identify patient with missing data and get the imputed value

```
pred.B <- predict(e.lmm, newdata = dtL.BNA, type = "dynamic",
    keep.newdata = TRUE)
pred.B$mdd <- paste(pred.B$mdd," (imputed)")
predOracle.B <- predict(eOracle.lmm, newdata = dtL.BNA, type = "dynamic",
   keep.newdata = TRUE)
predOracle.B$mdd <- paste(predOracle.B$mdd," (imputed)")
```

```
gg.imp <- ggplot(mapping = aes(x=time, color = group2))
gg.imp <- gg.imp + geom_boxplot(data = dtL.BNNA, mapping = aes(y = Yobs))
gg.imp <- gg.imp + geom_boxplot(data = dtL.BNA, mapping = aes(y = Yobs))
gg.imp <- gg.imp + geom_boxplot(data = pred.B, mapping = aes(y = estimate)
    )
gg.imp <- gg.imp + scale_color_manual("MDD group",
          values = c("limegreen","darkgreen","orange","red"))
gg.imp <- gg.imp + theme(text = element_text(size=15),
    axis.line = element_line(size = 1.25),
    axis.ticks = element_line(size = 2),
    axis.ticks.length=unit(.25, "cm"),
    legend.position="bottom",
    legend.direction = "horizontal")
gg.imp
```

```
Advarselsbeskeder:
1: Removed 969 rows containing non-finite values (stat_boxplot).
2: Removed 969 rows containing non-finite values (stat_boxplot).
```

```
gg.impOracle <- ggplot(mapping = aes(x=time, color = group2))
gg.impOracle <- gg.impOracle + geom_boxplot(data = dtL.BNNA, mapping = aes
    (y = Yobs))
gg.impOracle <- gg.impOracle + geom_boxplot(data = dtL.BNA, mapping = aes(
    y = Yobs))
gg.impOracle <- gg.impOracle + geom_boxplot(data = predOracle.B, mapping =
     aes(y = estimate))
gg.impOracle <- gg.impOracle + scale_color_manual("MDD group",
          values = c("limegreen","darkgreen","orange","red"))
gg.impOracle <- gg.impOracle + theme(text = element_text(size=15),
    axis.line = element_line(size = 1.25),
    axis.ticks = element_line(size = 2),
    axis.ticks.length=unit(.25, "cm"),
    legend.position="bottom",
    legend.direction = "horizontal")
gg.impOracle
```

```
Advarselsbeskeder:
1: Removed 969 rows containing non-finite values (stat_boxplot).
2: Removed 969 rows containing non-finite values (stat_boxplot).
```