Inverse probability of censoring weighting (IPCW) for linear regression

Brice Ozenne

August 3, 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.

2 Illustrative example

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 they are severaey depressed.

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

We can simulate such a dataset using the following function:

```
simTrial <- function(n, rho, dmu, pC){
  ## simulate data
  Sigma <- 10^2*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])
  )
  dtL$probaDO <- 0
  dtL[time=="T2", probaDO := ifelse(.SD$mdd=="moderate",pC[1],pC[2])]
  dtL[,dropout := rbinom(.N,prob=probaDO,size=1)]
  dtL[,Yobs:=Y]
  dtL[dropout==1,Yobs:=NA]
  dtL$probaDO <- NULL
  return(dtL)
}
set.seed(11)
dtL <- simTrial(n = 1000, rho = 0.8, dmu = c(25,50), pC = c(0.2,0.7))
print(dtL)
```

```
        id      mdd time        Y dropout     Yobs
   1:    1 moderate   T1 44.83259       0 44.83259
   2:    2 moderate   T1 30.34157       0 30.34157
   3:    3 moderate   T1 56.36308       0 56.36308
   4:    4 moderate   T1 64.63341       0 64.63341
   5:    5 moderate   T1 45.10048       0 45.10048
  ---
3996: 1996   severe   T2 30.59793       1       NA
3997: 1997   severe   T2 18.97725       1       NA
3998: 1998   severe   T2 29.80266       1       NA
3999: 1999   severe   T2 30.26518       0 30.26518
4000: 2000   severe   T2 39.15797       0 39.15797
```

Here we have simulated a two sub-populations of 1000, with a correlation of 0.5 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. Overall the expected treatment effect is:

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

```
[1] -37.5
```

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

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

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -37.35098  0.3141814 -118.8835        0
```

leading to an estimate quite close to the true value.

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

```
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) -31.42356  0.3909029 -80.38713        0
```

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

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

```
               Value Std.Error   DF   t-value p-value
(Intercept)  62.59128 0.3239587 1999 193.20760       0
timeT2      -33.76472 0.3855964 1068 -87.56494       0
```

which is better than the complete case analysis still biased when the drop-out mechanism depends on variables other than the baseline value.

A better approach is to use IPCW. First we model the probability of not dropping out at follow-up:

```
dtL$observed <- 1-dtL$dropout
dtL.T2 <- dtL[time == "T2"]
e.glmW.oracle <- glm(observed ~ mdd, data = dtL.T2,
       family = binomial(link = "logit"))
```

and then compute the weights for observations with full data:

```
w.oracle <- 1/predict(e.glmW.oracle, newdata = dtL.T2[dropout == 0],
        type = "response")
sum(w.oracle)
```

```
[1] 2000
```

Note that the weights sum to the total sample size. We then add these weights in our dataset:

```
dtW.CC$weights.oracle <- w.oracle
e.IPCWoracle <- lm(diff~1, data = dtW.CC, weights = dtW.CC$weights.oracle)
summary(e.IPCWoracle)$coef
```

```
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) -36.89889  0.4251421 -86.7919        0
```

which gives a result much closer to the true value.

3 Simulation study

The quality of the previous estimators is compared using a simulation study. The results are summarized by Figure 1.

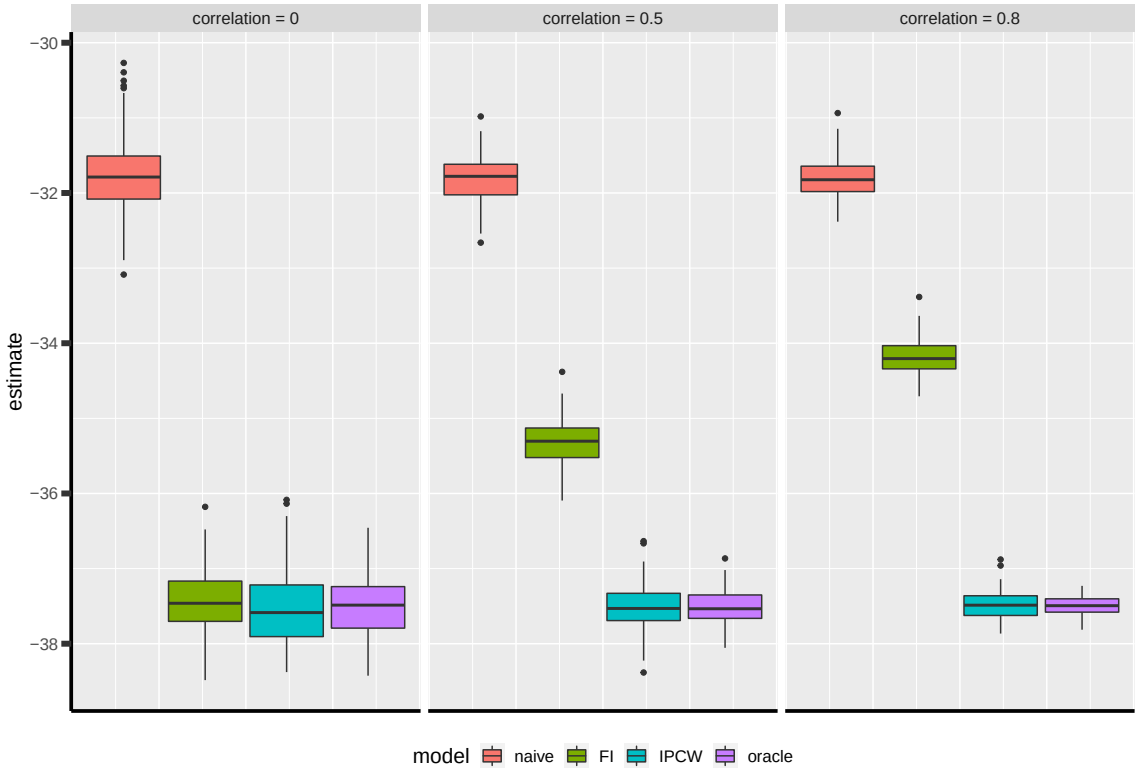

Figure 1: Results of the simulation study for a sample size of 1000 using 100 datasets.