

logistic_regression

2022-07-31

2) Complete data model is logistic regression & probability to be missing depends
only on Y

WTS: If probability to be missing depends ONLY on Y and NOT on X, and missing

data is confined to either Y or X (will confine to just X in our case), then

LD is more efficient (and generates unbiased regression coefficients)

```
create.data <- function(beta_0 = 0, beta_1 = 1, beta_2 = 2, n = 200){  
  # Data. Given this.  
  x1 = rnorm(n)          # some continuous variables  
  x2 = rnorm(n)  
  # defining z to be this. Coefficients are unknown. They are estimated using MLE as part of  
  # logistic regression.  
  z = beta_0 + beta_1*x1 + beta_2*x2      # linear combination with a bias. Here,  
  # B_0 = 1, B_1 = 2. The coefficients represent change in log odds. I.e. if  
  # x2 increases by 2, log odds increase by 2, i.e. odds of a "1" if x2 increases  
  # by 2 is exp(2) = 7.38 higher than original x2.  
  pr = 1/(1+exp(-z))      # pass through an inv-logit (sigmoid) function to  
                          # to constrain to [0,1]; represents odds of event  
                          # occurring  
  Y = rbinom(n,1,pr)  
  data = as.data.frame(cbind(Y, x1, x2))  
}
```

relative to multiple imputation.

Add Missingness

```
# probability of missingness is greater for x1/x2 if Y == 1.  
MNAR.make.missing <- function(data, prob_missing_larger = 0.2,  
                               prob_missing_smaller = 0.6){  
  # Setting up the randomness categories for missingness in x1, x2,  
  higher <- data %>% filter(Y == 1) %>% select(x1)  
  rx1_larger <- rbinom(nrow(higher), 1, prob_missing_larger)  
  rx1_smaller <- rbinom(nrow(data) - nrow(higher), 1, prob_missing_smaller)  
  rx2_larger <- rbinom(nrow(higher), 1, prob_missing_larger)  
  rx2_smaller <- rbinom(nrow(data) - nrow(higher), 1, prob_missing_smaller)  
  rx1 <- c(rx1_larger, rx1_smaller)  
  rx2 <- c(rx2_larger, rx2_smaller)  
  data <- data %>%
```

```

  arrange(desc(Y)) %>%
  cbind(rx1, rx2)
# Implementing the missingness in x1, X_3, X_4
data <- data %>% mutate(x1 = case_when(rx1 == 1 ~ as.numeric(NA),
                                     rx1 == 0 ~ as.numeric(data$x1))) %>%
  mutate(x2 = case_when(rx2 == 1 ~ as.numeric(NA),
                       rx2 == 0 ~ as.numeric(data$x2)))
# Remove setup variables
data <- select(data, -c(rx1, rx2))
data
}

```

Multiple Imputation

```

# Simulate multiple imputation, obtaining estimates and 95% confidence interval.
simulate_MI2 <- function(runs = 100) {
  res <- array(NA, dim = c(3, runs, 3))
  times <- array(NA, dim = c(runs, 1, 1))
  dimnames(res) <- list(c("Intercept", "x1", "x2"),
                       as.character(1:runs), c("estimate", "2.5%", "97.5%"))
  sim_dataset <- as.data.frame(create.data(n = 1000))
  for (run in 1:runs){
    # Note that time is only measured for the MI/imp steps
    # (i.e. filtering, predicting)
    missingness_sim_dataset <- MNAR.make.missing(sim_dataset, 0.2, 0.8)
    start_time <- Sys.time()
    imp_MI <- mice(missingness_sim_dataset, print = FALSE)
    fit <- with(imp_MI, glm(Y ~ x1 + x2, family = "binomial"))
    end_time <- Sys.time()
    tab <- summary(pool(fit), "all", conf.int = TRUE)
    res[1, run, ] <- as.numeric(tab[1, c("estimate", "2.5 %", "97.5 %")])
    res[2, run, ] <- as.numeric(tab[2, c("estimate", "2.5 %", "97.5 %")])
    res[3, run, ] <- as.numeric(tab[3, c("estimate", "2.5 %", "97.5 %")])

    times[run, 1, 1] <- as.numeric(end_time - start_time)
  }
  list(res, times)
}

```

```

# Run 100 iterations
res_MI2 <- simulate_MI2(100)

```

```

# Obtain confidence intervals & estimates for all coefficients, intercept.
apply(res_MI2[[1]], c(1, 3), mean, na.rm = TRUE)

```

```

##           estimate      2.5%      97.5%
## Intercept 0.07608593 -0.3269558 0.4791277
## x1        1.11169267  0.3876110 1.8357743
## x2        2.21261059  1.3259445 3.0992766

```

```

# Mean time for the multiple imputation instances
times <- res_MI2[[2]]
mean(times)

```

```

## [1] 0.2123698

```

```

# Evaluating imputation method performance for estimating
# all parameters of interest.
res <- res_MI2[[1]]
true <- c(0, 1, 2)
RB <- rowMeans(res[, "estimate"]) - true
PB <- 100 * abs((rowMeans(res[, "estimate"]) - true) / true)
CR <- rowMeans(res[, "2.5%"] < true & true < res[, "97.5%"])
AW <- rowMeans(res[, "97.5%"] - res[, "2.5%"])
RMSE <- sqrt(rowMeans((res[, "estimate"] - true)^2))
data.frame(RB, PB, CR, AW, RMSE)

```

```

##           RB      PB  CR      AW      RMSE
## Intercept 0.07608593      Inf 0.99 0.8060835 0.1511027
## x1        0.11169267 11.16927 0.98 1.4481633 0.2403467
## x2        0.21261059 10.63053 0.98 1.7733321 0.3379104

```

Listwise Deletion

```

# Simulate listwise deletion, obtaining estimates and 95% confidence interval.

simulate_LD <- function(runs = 100){
  res <- array(NA, dim = c(3, runs, 3))
  dimnames(res) <- list(c("Intercept", "x1", "x2"),
                        as.character(1:runs), c("estimate", "2.5%", "97.5%"))
  times <- array(NA, dim = c(runs, 1, 1))
  sim_dataset <- as.data.frame(create.data(n = 1000))
  # Note that time is only measured for the LD/imp steps (i.e. filtering, predicting)
  for (run in 1:runs){
    missingness_sim_dataset <- MNAR.make.missing(sim_dataset, 0.2, 0.5)
    start_time <- Sys.time()
    filtered_sim_dataset <- missingness_sim_dataset %>%
      select(Y, x1, x2) %>%
      filter(!is.na(x1), !is.na(x2))

    fit <- with(filtered_sim_dataset, glm(Y ~ x1 + x2, family = "binomial"))
    end_time <- Sys.time()
    times[run, 1, 1] <- as.numeric(end_time - start_time)
    # loop over each variable. Note we do the imputation just ONCE b/c LD is
    # deterministic.
    for (var in 1:3){
      edges <- as.numeric((confint.default(fit))[var,])
      estimate <- as.numeric(fit$coefficients)[var]
      interval <- c(estimate, edges)
      res[var, run, ] <- interval
    }
  }
  list(res, times)
}

result_LD <- simulate_LD(100)

```

```

# Obtain confidence intervals & estimates for all coefficients, intercept.
apply(result_LD[[1]], c(1, 3), mean, na.rm = TRUE)

```

```
##           estimate      2.5%    97.5%
## Intercept 0.9215474 0.6442881 1.198807
## x1        1.1480406 0.8234423 1.472639
## x2        2.0521936 1.6225424 2.481845

# Evaluating imputation method performance for estimating
# all parameters of interest.
res <- result_LD[[1]]
true <- c(0, 1, 2)
RB <- rowMeans(res[, "estimate"]) - true
PB <- 100 * abs((rowMeans(res[, "estimate"]) - true) / true)
CR <- rowMeans(res[, "2.5%"] < true & true < res[, "97.5%"])
AW <- rowMeans(res[, "97.5%"] - res[, "2.5%"])
RMSE <- sqrt(rowMeans((res[, "estimate"] - true)^2))
data.frame(RB, PB, CR, AW, RMSE)

##           RB          PB    CR          AW    RMSE
## Intercept 0.92154738      Inf 0.00 0.5545185 0.9290863
## x1        0.14804061 14.804061 0.96 0.6491966 0.1979282
## x2        0.05219365  2.609682 1.00 0.8593025 0.1655004

# Mean time for 100 instances of LD
times_LD <- result_LD[[2]]
mean(times_LD)

## [1] 0.007046299
```