How to perform parameters estimation with missing values?

Pavlo Mozharovskyi, Wei Jiang, Manuel Pichon

19 January 2021

Contents

Imagine you are solving a linear regression but there are a lot of missing values in your covariate matrix, how can you deal with so many "*NA*"? The trouble may be caused by survey non-responses, lost records or machine failures. Classical packages performing estimation would let you ignore all the lines with missingness, as a result, much information would be lost.

A more reasonable method to handle this problem, is modifying an estimation process so that the method can be applied to incomplete data. For example, one can use the Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) to obtain the maximum likelihood estimate (MLE) despite missing values. This strategy is valid under missing at random (MAR) mechanisms (Little and Rubin 2002; Seaman et al. 2013), in which the missingness of data is independent of the missing values, given the observed data. Even though this approach is perfectly suited to specific inference problems with missing values, there are few solutions or implementations available, even for simple models. In the following , we will demonstrate how to estimate parameters based on EM algorithm in a linear regression model and in a logistic regression model.

Another popular choice is multiple imputation (Buuren and Groothuis-Oudshoorn 2011), followed by classical regression procedure on each imputed dataset, and finally we aggregate the estimate on each set with Rubin's combining rules. In the following, we will compare this method to EM and illustrate the bias and variance of estimation by an example of simulated dataset.

Linear regression with missing values

Estimation with EM algorithm

Consider the following linear regression model which fits to numerous practical settings even while being restricted to the joint multivariate normal assumption:

$$y = \tilde{X}^\top \boldsymbol{\beta} + \varepsilon \,,$$

where $\tilde{X} = (1, X^\top)^\top$ with random variable $X = (x_1, ..., x_p)^\top \sim \mathcal{N}_p(\mu_X, \Sigma_X)$ independent of the noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Let n *i.i.d.* observations $(y_i, X_i^\top)^\top$, $i = 1, ..., n$ be available but only partly observed. We aim to estimate $\mathbb{E}[y \mid X] = \tilde{X}^\top \boldsymbol{\beta}$.

In view of independence and normality of y and X, we have

$$(y, X) \sim \mathcal{N}(\mu_{y,X}, \Sigma_{y,X}) \quad \text{with} \quad \mu_{y,X} = \begin{pmatrix} \mu_y \\ \mu_X \end{pmatrix} \quad \text{and} \quad \Sigma_{y,X} = \begin{pmatrix} \Sigma_y & \Sigma_{y,X} \\ \Sigma_{X,y} & \Sigma_X \end{pmatrix} .$$

Let $\theta = (\mu_{y,X}, \Sigma_{y,X})$ denote a set of parameters. θ can be estimated using the EM algorithm (Dempster, Laird, and Rubin 1977) and this presentation allows us to write

$$\mathbb{E}[y \mid X] = (\mu_y - \Sigma_{y,X}\Sigma_X^{-1}\mu_X) + \Sigma_{y,X}\Sigma_X^{-1}X \,,$$

and thus $\boldsymbol{\beta}$ can be estimated by plug-in from

$$\boldsymbol{\beta} = (\mu_y - \Sigma_{y,X}\Sigma_X^{-1}\mu_X, \Sigma_{y,X}\Sigma_X^{-1})^\top \,.$$

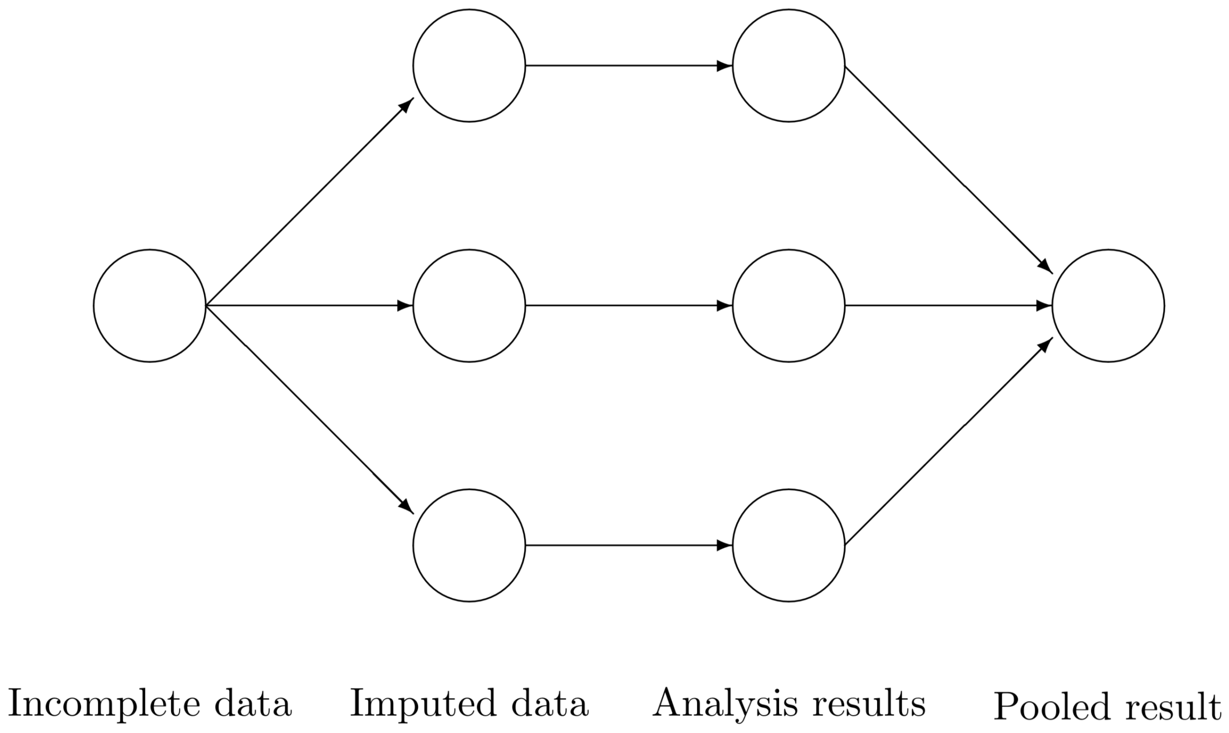
In the same way, the standard deviations can be estimated (via the Gram matrix) as

$$\mathbb{V}[\boldsymbol{\beta}] = \text{diag}(C) \quad \text{with} C = (\Sigma_y - \boldsymbol{\beta}^\top \Sigma_X \boldsymbol{\beta})\big((\mathbf{0}_{p+1}, (\mathbf{0}_p, \Sigma_X)^\top)^\top + (1, \mu_X^\top)^\top (1, \mu_X^\top)\big)^{-1}/n \,.$$

Estimation via multiple imputation (MI)

Overview of MI framework

Multiple imputation creates $M > 1$ complete datasets, and then a parameter of interest θ can be estimated from each imputed dataset. This second step is performed by applying the analytic method we would have used had the data been complete. Let's denote by $\hat{\theta}_k$ the k^{th} completed-data estimate of θ (with $k \in \{1, \ldots, M\}$), and its estimated variance $\hat{V}_k$. Depending on the values imputed in the k^{th} completed dataset, the result for each estimated parameter $\hat{\theta}_k$ will obviously differ. The third and final pooling step is performed using specific rules named "Rubin's rules" (Rubin 1987) that will led to a final point estimate. The figure below illustrates the main steps described above :



Rubin's rules

Let's denote by $\bar{\theta}_{MI}$ the pooled estimator of θ. It's obtained by taking the average over the parameter estimates $\hat{\theta}_k$ from all M imputed datasets :

$$\bar{\theta}_{MI} = \frac{1}{M}\sum_{k=1}^{M}\hat{\theta}_k$$

Let's denote by $\overline{V}_{MI}$ the pooled estimated variance of $\bar{\theta}_{MI}$. It's obtained by combining the within-imputation component $\overline{V}_{within}$ and between-imputation component $\overline{V}_{between}$ of overall variance as follows :

$$\begin{array}{ccccc} \overline{V}_{MI} & = & \overline{V}_{within} & + & \left(1+\frac{1}{M}\right)\overline{V}_{between} \\ & = & \frac{1}{M}\sum_{k=1}^{M}\hat{V}_k & + & \left(1+\frac{1}{M}\right)\cdot\frac{1}{M-1}\sum_{k=1}^{M}(\hat{\theta}_k-\bar{\theta}_{MI})(\hat{\theta}_k-\bar{\theta}_{MI})^\top \end{array}$$

Example on a simulated dataset

This idea is implemented in the following chunk of code.

```
require(MASS)
```

```
## Loading required package: MASS
```

```
require(norm)
```

```
## Loading required package: norm
```

```
require(mice)
```

```
## Loading required package: mice
```

```
##
## Attaching package: 'mice'
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
set.seed(1)
# Sample data generation ------------------------------------------------------
# Generate complete data
mu.X <- c(1, 1)
Sigma.X <- matrix(c(1, 1, 1, 4), nrow = 2)
n <- 1000
X.complete <- mvrnorm(n, mu.X, Sigma.X)
b <- c(2, 3, -1)
sigma.eps <- 0.25
y.complete <- cbind(rep(1, n), X.complete) %*% b + rnorm(n, 0, sigma.eps)
reg.complete <- lm(y.complete ~ X.complete)
summary(reg.complete)
```

```
##
## Call:
## lm(formula = y.complete ~ X.complete)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90377 -0.16411 -0.00558  0.17037  0.70330
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.001088   0.011366   176.1   <2e-16 ***
## X.complete1  2.995404   0.009059   330.7   <2e-16 ***
## X.complete2 -0.992432   0.004526  -219.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2577 on 997 degrees of freedom
## Multiple R-squared:  0.9913, Adjusted R-squared:  0.9913
## F-statistic: 5.674e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Add missing values
yX.miss <- ampute(cbind(y.complete, X.complete), 0.15, patterns = matrix(
  c(0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0),
  ncol = 3, byrow = TRUE), freq = c(1, 1, 1, 2, 2, 2) / 9,
  mech = "MCAR", bycases = FALSE)
y <- yX.miss$amp[,1]              # OUTPUT of the linear regression with NAs
X <- as.matrix(yX.miss$amp[,2:3]) # INPUT  of the linear regression with NAs
# Estimation of the regression using EM----------------------------------------
# Estimate extended mean and covariance using EM
s <- prelim.norm(cbind(y, X))
thetahat <- em.norm(s)
```

```
## Iterations of EM:
## 1...2...3...4...5...6...7...8...9...10...
```

```
pars <- getparam.norm(s, thetahat)
# Calculate regression estimates
b.est <- c(pars$mu[1] -
             pars$sigma[1,2:3] %*% solve(pars$sigma[2:3,2:3]) %*% pars$mu[2:3],
```

```r
           pars$sigma[1,2:3] %*% solve(pars$sigma[2:3,2:3]))
esigma.est <- as.vector(sqrt(
  pars$sigma[1,1] - b.est[2:3] %*% pars$sigma[2:3,2:3] %*% b.est[2:3]))
Gram.est <- rbind(rep(0, 3), cbind(rep(0, 2), pars$sigma[2:3,2:3])) +
  c(1, pars$mu[2:3]) %*% t(c(1, pars$mu[2:3]))
sdb.est <- sqrt(diag(esigma.est^2 * solve(Gram.est * n)))
cat("Estimated regression coefficients:", b.est, "\n")
```

```
## Estimated regression coefficients: 2.00462 2.990707 -0.9941848
```

```r
cat("Their standard deviations:", sdb.est, "\n")
```

```
## Their standard deviations: 0.01147407 0.009113964 0.004496219
```

```r
cat("Standard deviation of residuals:", esigma.est, "\n")
```

```
## Standard deviation of residuals: 0.2593369
```

```r
# Estimation of the regression using multiple imputation ----------------------
# Using R-package "mice"
mice.est <- mice(cbind(y, X), maxit = 5, m = 20, printFlag = FALSE) # impute
fit <- with(data = mice.est, exp = lm(y ~ V2 + V3)) # fit all m models
mice.lmodel <- mice::pool(fit) # pool the results using Rubin's rules
summary(mice.lmodel)
```

```
##          term   estimate  std.error statistic       df p.value
## 1 (Intercept)  2.0156169 0.01432684  140.6882 164.1600       0
## 2          V2  2.9793208 0.01184887  251.4435 117.6911       0
## 3          V3 -0.9885609 0.00574563 -172.0544 132.8897       0
```

Logistic regression with missing values

Logistic regression model

Let (y_i, X_i) be n $i.i.d$ observations with $y_i \in \{0,1\}$ a binary response and $X_i = (X_{i1}, \dots, X_{ip})^\top \in \mathbb{R}^p$ vector of covariates. The logistic regression model for binary classification can be written as:

$$\mathbb{P}(y_i = 1 \mid X_i; \beta) = \frac{\exp(\beta_0 + \sum_{j=1}^p \beta_j X_{ij})}{1 + \exp(\beta_0 + \sum_{j=1}^p \beta_j X_{ij})}, \quad i = 1, \dots, n,$$

where $\beta = (\beta_0, \beta_1, \dots, \beta_p)^\top$ are unknown parameters. We adopt a probabilistic framework by assuming that X_i is normally distributed:

$$X_i \underset{\text{i.i.d.}}{\sim} \mathcal{N}_p(\mu, \Sigma), \quad i = 1, \cdots, n.$$

Let $\theta = (\mu, \Sigma, \beta)$ be the set of parameters of the model. Then, the log-likelihood for the complete data can be written as:

$$\ell(\theta; X, y) = \sum_{i=1}^n \ell(\theta; X_i, y_i) = \sum_{i=1}^n \left(\log \mathrm{p}(y_i \mid X_i; \beta) + \log \mathrm{p}(X_i; \mu, \Sigma)\right).$$

where $X = \begin{pmatrix} X_1^\top \\ X_2^\top \\ \vdots \\ X_n^\top \end{pmatrix}$ the design matrix and $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$ vector of response. Our main goal is to estimate the vector of parameters β when missing values (MAR) exist in the design matrix: $X = (X_{\text{OBS}}, X_{\text{MIS}})$.

Parameter estimation by a stochatic approximation version of EM

For logistic regression with EM algorithm, there is no explicit expression to calculate expectation in E step. Therefore, a Monte Carlo version of EM (Ibrahim, Chen, and Lipsitz 1999) can be used. The E-step of MCEM generates a large number of samples of missing data from the target distribution $\mathtt{p}(X_{\text{MIS}}|X_{\text{OBS}}, y; \theta)$ and replaces the expectation of the complete log-likelihood by an empirical mean. However, an accurate Monte Carlo approximation of the E-step may require a significant computational effort. To achieve improved computational efficiency, Stochastic Approximation EM (SAEM) algorithm (Lavielle 2014) replaces the E-step by a stochastic approximation based on a single simulation of X_{MIS}. Starting from an initial guess $\theta^{(0)}$, the tth iteration consists of three steps:

- Simulation: For $i = 1, 2, \cdots, n$, draw a single sample $X_{\text{MIS}}^{(t)}$ from the conditional distribution of missing variables: $\mathtt{p}(X_{\text{MIS}} \mid X_{\text{OBS}}, y; \theta^{(t-1)})$.
- Stochastic approximation: Update the function Q, *i.e.*, the expected complete-data log-likelihood, according to
$$Q(\theta, \theta^{(t)}) = Q(\theta, \theta^{(t-1)}) + \gamma_t \left(\ell(\theta; X_{\text{OBS}}, X_{\text{MIS}}^{(t)}, y) - Q(\theta, \theta^{(t-1)}) \right),$$
where (γ_t) is a decreasing sequence of positive numbers.
- Maximization: update the estimation of θ:

$$\theta^{(t+1)} = argmax_\theta Q(\theta, \theta^{(t)}).$$

Example on a simulated dataset

Parameter estimation with SAEM

The methodology is implemented in the package misaem. Let's illustrate the use of the package on a simulated data.

We first generate a design matrix of size $N = 500$ times $p = 5$ by drawing each observation from a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. Then, we generate the response according to the logistic regression model.

We consider as the true values for the parameters

$$\begin{aligned} \beta &= (0, 1, -1, 1, 0, -1)^\top, \\ \mu &= (1, 2, 3, 4, 5)^\top, \\ \Sigma &= \text{diag}(\sigma) C \text{diag}(\sigma), \end{aligned}$$

where the vector of standard deviations $\sigma = (1, 2, 3, 4, 5)^\top$, and the correlation matrix:

$$C = \begin{bmatrix} 1 & 0.8 & 0 & 0 & 0 \\ 0.8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.3 & 0.6 \\ 0 & 0 & 0.3 & 1 & 0.7 \\ 0 & 0 & 0.6 & 0.7 & 1 \end{bmatrix}.$$

```r
# Generate dataset
set.seed(200)
N <- 500  # number of subjects
p <- 5     # number of explanatory variables
mu.star <- 1:p  #rep(0,p)  # mean of the explanatory variables
sd <- 1:p # rep(1,p) # standard deviations
C <- matrix(c(   # correlation matrix
1,   0.8, 0,   0,   0,
0.8, 1,   0,   0,   0,
0,   0,   1,   0.3, 0.6,
0,   0,   0.3, 1,   0.7,
0,   0,   0.6, 0.7, 1), nrow=p)
Sigma.star <- diag(sd)%*%C%*%diag(sd) # covariance matrix
beta.star <- c(1, -1, 1, 0, -1) # coefficients
beta0.star <- 0  # intercept
beta.true = c(beta0.star,beta.star)

# Design matrix
X.complete <- matrix(rnorm(N*p), nrow=N)%*%chol(Sigma.star)+
              matrix(rep(mu.star,N), nrow=N, byrow = TRUE)

# Reponse vector
p1 <- 1/(1+exp(-X.complete%*%beta.star-beta0.star))
y <- as.numeric(runif(N)<p1)
```

Then we randomly introduced 10% of missing values in the covariates according to the MCAR (Missing completely at random) mechanism.

```r
# Generate missingness
set.seed(200)
p.miss <- 0.10
patterns <- runif(N*p)<p.miss # missing completely at random
X.obs <- X.complete
X.obs[patterns] <- NA
```

Have a look at our synthetic dataset:

```r
head(X.obs)
```

```
##           [,1]        [,2]      [,3]         [,4]        [,5]
## [1,] 1.0847563  1.71119812 5.0779956  9.731254821 13.02285225
## [2,] 1.2264603  0.04664033 5.3758000  6.383093558  4.84730504
## [3,] 1.4325565  1.77934455        NA  8.421927692  7.26902254
## [4,] 1.5580652  5.69782193 5.5942869 -0.440749372 -0.96662931
## [5,] 1.0597553 -0.38470918 0.4462986  0.008402997  0.04745022
## [6,] 0.8853591  0.56839374 3.4641522  7.047389616          NA
```

The main function in package is `miss.saem` function, which returns the estimation of parameters for logistic regression with missingness. Here we apply this function with its default options.

```r
# Charge library
# install.packages("misaem")
library(misaem)

# SAEM
df.obs = data.frame(y, X.obs)
miss.list = miss.glm(y~., data=df.obs, print_iter = FALSE, seed=100)

# Summary
print(summary(miss.list))
```

```
##
## Call:
## miss.glm(formula = y ~ ., data = df.obs, print_iter = FALSE,
##     seed = 100)
##
## Coefficients:
```

```
##              Estimate  Std. Error
## (Intercept)   0.05128   0.31942
## X1            1.05798   0.35989
## X2           -0.99347   0.19620
## X3            1.07606   0.13937
## X4           -0.02258   0.06604
## X5           -1.01527   0.13353
## Log-likelihood: -132.14
```

```
# estimation of parameters
print(miss.list$coefficients)
```

```
## (Intercept)          X1          X2          X3          X4          X5
##  0.05127840  1.05798244 -0.99346626  1.07605641 -0.02257848 -1.01527374
```

And if you need to obtain the variance of estimation:

```
# estimation of variance
print(miss.list$var.covar)
```

```
##              [,1]         [,2]          [,3]         [,4]          [,5]
## [1,]  0.102030117 -0.016029956 -7.885070e-03 -0.008158977 -6.347421e-03
## [2,] -0.016029956  0.129524244 -5.936861e-02  0.009260344 -1.109363e-03
## [3,] -0.007885070 -0.059368607  3.849495e-02 -0.011084240  8.359573e-05
## [4,] -0.008158977  0.009260344 -1.108424e-02  0.019424653  2.309738e-03
## [5,] -0.006347421 -0.001109363  8.359573e-05  0.002309738  4.361918e-03
## [6,]  0.001214081 -0.010280996  1.086142e-02 -0.016592695 -4.051082e-03
##              [,6]
## [1,]  0.001214081
## [2,] -0.010280996
## [3,]  0.010861419
## [4,] -0.016592695
## [5,] -0.004051082
## [6,]  0.017829642
```

Parameter estimation with Multiple Imputation (MI)

The purpose of this part is to give an example of parameter estimation using multiple imputation in the logistic regression case, performed on the same simulated dataset as used above in the previous section. As a reminder, the MI framework is described in the linear regression section.

We recall that we want to provide an estimate of the parameter β given by `beta.true` as follows :

```
beta.true
```

```
## [1]  0  1 -1  1  0 -1
```

We'll use multiple imputation on our incomplete dataset `X.obs` to perform the estimation task.

We use the package `mice` and first impute values to generate 20 complete datasets :

```
library(mice)
mi <- mice(data.frame(y, X.obs), m=20, printFlag = FALSE)  # imputation of 20 complete datasets
```

One can observe below an overview of the 2 first completed datasets, where missing values in `X.obs[3,3]` and `X.obs[6,5]` have basically been replaced by plausible values.

```
complete.dataset.1 <- complete(mi, action=1) # completed dataset #1
head(complete.dataset.1)
```

```
##   y        X1          X2          X3           X4          X5
## 1 0 1.0847563  1.71119812  5.07799564  9.731254821 13.02285225
## 2 1 1.2264603  0.04664033  5.37579999  6.383093558  4.84730504
## 3 0 1.4325565  1.77934455 -0.04566088  8.421927692  7.26902254
## 4 1 1.5580652  5.69782193  5.59428692 -0.440749372 -0.96662931
## 5 0 1.0597553 -0.38470918  0.44629862  0.008402997  0.04745022
## 6 0 0.8853591  0.56839374  3.46415220  7.047389616 10.50540253
```

```
complete.dataset.2 <- complete(mi, action=2) # completed dataset #2
head(complete.dataset.2)
```

```
##   y        X1          X2        X3           X4          X5
## 1 0 1.0847563  1.71119812 5.0779956  9.731254821 13.02285225
## 2 1 1.2264603  0.04664033 5.3758000  6.383093558  4.84730504
## 3 0 1.4325565  1.77934455 2.5954812  8.421927692  7.26902254
## 4 1 1.5580652  5.69782193 5.5942869 -0.440749372 -0.96662931
## 5 0 1.0597553 -0.38470918 0.4462986  0.008402997  0.04745022
## 6 0 0.8853591  0.56839374 3.4641522  7.047389616  7.62533573
```

The estimation task is then computed on each completed dataset using the function `with`, and the pooled estimates are finally obtained through the use of `pool`:

```
fit <- with(data = mi, exp = glm(y ~ X1+X2+X3+X4+X5, family = binomial)) # fit
beta.mi <- mice::pool(fit) # pool the results using Rubin's rules
summary(beta.mi)
```

```
##          term    estimate  std.error  statistic        df      p.value
## 1 (Intercept)  0.04006508 0.32034287  0.1250694 325.01965 9.005460e-01
## 2          X1  0.85919319 0.35413092  2.4262021 178.92213 1.625036e-02
## 3          X2 -0.85098985 0.19626265 -4.3359745 123.30329 2.983626e-05
## 4          X3  0.99568077 0.14825886  6.7158263  85.76393 1.934425e-09
## 5          X4 -0.04100766 0.06938153 -0.5910457 126.65685 5.555431e-01
## 6          X5 -0.92834313 0.14636424 -6.3426908  65.36987 2.423562e-08
```

Parameter estimation on complete dataset

For comparison purpose, we compute hereafter an estimate of β using our complete dataset `X.complete`

```
beta.complete <- glm(y ~ X.complete, family = binomial())
summary(beta.complete)
```

```
##
## Call:
## glm(formula = y ~ X.complete, family = binomial())
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.07303  -0.31332  -0.06349   0.07456   3.16145
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.008648   0.307054   0.028 0.977532
## X.complete1  1.121390   0.325521   3.445 0.000571 ***
## X.complete2 -1.007353   0.173942  -5.791 6.98e-09 ***
## X.complete3  1.097109   0.126773   8.654  < 2e-16 ***
## X.complete4 -0.037535   0.058054  -0.647 0.517919
## X.complete5 -1.036672   0.121119  -8.559  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 587.20  on 499  degrees of freedom
## Residual deviance: 230.85  on 494  degrees of freedom
## AIC: 242.85
##
## Number of Fisher Scoring iterations: 7
```

Session info

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS  10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] misaem_1.0.0  mice_3.11.0   norm_1.0-9.5  MASS_7.3-51.6
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.5       knitr_1.29       magrittr_2.0.1   tidyselect_1.1.0
##  [5] lattice_0.20-41  R6_2.5.0         rlang_0.4.10     stringr_1.4.0
##  [9] dplyr_1.0.2      tools_4.0.2      grid_4.0.2       broom_0.7.0
## [13] xfun_0.19        htmltools_0.5.0  ellipsis_0.3.1   yaml_2.2.1
## [17] digest_0.6.27    tibble_3.0.4     lifecycle_0.2.0  crayon_1.3.4
## [21] tidyr_1.1.2      purrr_0.3.4      vctrs_0.3.6      glue_1.4.2
## [25] evaluate_0.14    rmarkdown_2.6    stringi_1.5.3    compiler_4.0.2
## [29] pillar_1.4.7     backports_1.1.9  generics_0.1.0   mvtnorm_1.1-1
## [33] pkgconfig_2.0.3
```

References

Buuren, Stef van, and Karin Groothuis-Oudshoorn. 2011. "mice: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software* 45 (3): 1–67. http://www.jstatsoft.org/v45/i03/.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. 1977. "Maximum Likelihood from Incomplete Data via the Em Algorithm." *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1): 1–38. http://www.jstor.org/stable/2984875.

Ibrahim, Joseph G., Ming-Hui Chen, and Stuart R. Lipsitz. 1999. "Monte Carlo Em for Missing Covariates in Parametric Regression Models." *BIOMETRICS* 55: 591–96.

Lavielle, Marc. 2014. *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*. Chapman and Hall/CRC. https://hal.archives-ouvertes.fr/hal-01122873.

Little, Roderick J. A., and Donald B. Rubin. 2002. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc.

Rubin, Donald B. 1987. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons.

Seaman, Shaun, John Galati, Dan Jackson, and John Carlin. 2013. "What Is Meant by 'Missing at Random'?" *Statist. Sci.* 28 (2): 257–68. https://doi.org/10.1214/13-STS415.