Introduction to clinical trials
Lab 7: Analyses in the presence of missing data

Constantin T Yiannoutsos & Christos Thomadakis

November 2, 2017

1 The missing data hierarchy

The levels difficulty of interpreting missing data in a model of response Y, modeled according to a set of predictors (also known as covariates) X, are given below (Little & Rubin, 1987):

- *MCAR*
  Data are missing completely at random. The missingness is not associated with the actual true (but uknown) value and there are no other variables in the data that can predict whether a missing value exists in a particular variable. This is the difference between MCAR and MAR (below).

- *MAR*
  Data are missing at random. The pattern or probability of missingness is associated with other variables in the data *but not with the actual missing value*. This is the difference between MCAR and MNAR (see below).

- *MNAR*
  Data are missing not at random. The chance that the data are missing depends on the actual missing value.

MCAR are data missing completely at random. In clinical trials measuring a response Y based on measurements obtained from the subjects (covariates) X.

MCAR means that missing responses and/or covariate measurements are not dependent upon other responses or covariate measurements.

An intermediate situation is missing data dependent only on values of the covariates X but not on Y. This is called covariate-dependent missingness by Little (1995, *JASA*). In this case, missing data are dependent only on the covariates X but not on the outcomes Y.

1.1 Example of covariate-dependent missingness

We first import the data in `R`. Recall that missing data are denoted as `NA` in `R`. The function `is.na` is a very useful function that tell us if an observation is missing.

```
reg =  read.csv("H:/teaching/Indiana/PBHL B-582/labs/lab 7 (data analysis I)/data/reg.csv")

# Have a look at the data
reg
```

```
##    unit   x    y
## 1     1 3.4 5.67
## 2     2 3.9 4.81
## 3     3 2.6 4.93
## 4     4 1.9 6.21
## 5     5 2.2 6.83
## 6     6 3.3 5.61
## 7     7 1.7 5.45
## 8     8 2.4 4.94
## 9     9 2.8 5.73
## 10   10 3.6   NA
```

```
is.na(reg)
```

```
##        unit     x     y
##  [1,] FALSE FALSE FALSE
##  [2,] FALSE FALSE FALSE
##  [3,] FALSE FALSE FALSE
##  [4,] FALSE FALSE FALSE
##  [5,] FALSE FALSE FALSE
##  [6,] FALSE FALSE FALSE
##  [7,] FALSE FALSE FALSE
##  [8,] FALSE FALSE FALSE
##  [9,] FALSE FALSE FALSE
## [10,] FALSE FALSE  TRUE
```

```
# To see the number of missing values
apply(reg,2, function(x) sum(is.na(x)) )
```

```
## unit    x    y
##    0    0    1
```

We see that the last observation of Y is missing while X contains no missing data. To fit a linear model we can use the function `lm`.

```
# We're interested in regressing Y on X
# Complete-case analysis (the default)
fitCC = lm(y ~ x, data = reg)
betaCC = coef(fitCC)
summary(fitCC)
```

```
##
```

```
## Call:
## lm(formula = y ~ x, data = reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7413 -0.4876  0.1951  0.3456  1.0754
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.5601     0.8565   7.660  0.00012 ***
## x            -0.3662     0.3085  -1.187  0.27399
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6389 on 7 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.1675,Adjusted R-squared:  0.0486
## F-statistic: 1.409 on 1 and 7 DF,  p-value: 0.274
```

Note that R excludes any missing data by default (as almost all packages do).

Now we move on the imputation of missing data. Assuming that the model based on the observed data is correct, we can impute the missing observation using the fitted model.

```
# We carry out a single imputation
reg.singImp = reg

# Indices of missing values
mis = is.na(reg.singImp$y)
reg.singImp$y[mis] = cbind(1,reg.singImp$x[mis]) %*% betaCC

# Imputed analysis with single imputation
fitSingImp = lm(y ~ x, data = reg.singImp)
summary(fitSingImp)

##
## Call:
## lm(formula = y ~ x, data = reg.singImp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.74134 -0.44626  0.09756  0.32374  1.07543
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.5601     0.7641   8.585 2.62e-05 ***
## x            -0.3662     0.2663  -1.375    0.206
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5977 on 8 degrees of freedom
## Multiple R-squared:  0.1911,Adjusted R-squared:  0.09002
## F-statistic:  1.89 on 1 and 8 DF,  p-value: 0.2064
```

Let's unpack the previous code. The operator `%*%` in the line

```
reg.singImp$y[mis] = cbind(1,reg.singImp$x[mis]) %*% betaCC
```

performs the inner product of the vector $\hat{\beta}' = (6.5601, -0.3662)$ with the row of the design matrix X which has the missing data (indexed by `[mis]`, $X_{10} = (1, 3.6)$). This is

$$(1, 3.6) \begin{pmatrix} 6.5601 \\ -0.3662 \end{pmatrix} = 6.5601 + 3.6(-0.3662) = 5.2419$$

We see that the estimated slope has remained the same. However, the standard error of the slope has decreased from 0.31 to 0.27. This illustrates the serious drawbacks of such an approach: it does not account for the fact that the missing observation is random!

One solution is to simulate the missing observation many times and fit the model on the complete data. However, to produce a reasonable standard error, we need to resample the dataset. The function `sample` is very useful in doing this.

```
regboot = function(data, N = 1000)
 {
   # Data frame including the results of each iteration
   results = data.frame(it = 1:N,betaHat0 = NA, betaHat1 = NA)

   # Number of observations
   n = nrow(data)

   for (i in 1:N)
   {
     # Bootstrap sample
     temp_data = data[sample(n,replace = T),]

     # Fit the model on the complete cases
     fit = lm(y ~ x, data = temp_data)
     beta = coef(fit)
     sd = summary(fit)$sigma

     # Indices of missing values
     mis = is.na(temp_data$y)
     if (any(mis))
     {
```

```
        # Impute the missing values; if any
        temp_data$y[mis] = cbind(1,temp_data$x[mis])%*%beta
        + rnorm(1,mean = 0,sd = sd)
      }

      fit = lm(y ~ x, data = temp_data)
      results[i,] = c(i,coef(fit))
    }
    return(results)
}
# Set the seed to a specific value
set.seed(1255)
sample.beta = regboot(reg)
```

This means that the estimate β = -0.345 with an attendant standard error $se(\beta) = 0.330$, which is much more realistic than the one produced by the naive imputation procedure described earlier.

Now let's unpack the code. The code fragment

```
regboot = function(data, N = 1000)
```

creates a new function `regboot`. This function has two arguments: `data` and `N=1000` the number of bootstrapped samples to be created.

The code

```
# Data frame including the results of each iteration
   results = data.frame(it = 1:N,betaHat0 = NA, betaHat1 = NA)
```

creates a blank data frame with $N = 1000$ rows, as many as the imputations we will carry out, where all the data will ultimately be stored. You need to do this in order to reserve space in the memory for the ultimate results.

The code fragment

```
# Bootstrap sample
     temp_data = data[sample(n,replace = T),]
```

resamples the data. By "resamples" we mean that it creates a new data set with the same number of rows as the one we are working with, by sampling *with replacement* from the *indices* (row numbers) of the original data set (which is passed to our newly defined function `regboot` by the argument `data`).

The code fragment

```
    # Indices of missing values
    mis = is.na(temp_data$y)
    if (any(mis))
    {
      # Impute the missing values; if any
      temp_data$y[mis] = cbind(1,temp_data$x[mis])%*%beta
      + rnorm(1,mean = 0,sd = sd)
    }
```

checks whether the line with the missing data was part of the resampled data set and imputes, in the manner outlined in the single-imputation above, the missing y. Note that, because of the resampling, not all resampled data sets will include the line with the missing data from the original data set.

```
      # Impute the missing values; if any
      temp_data$y[mis] = cbind(1,temp_data$x[mis])%*%beta
      + rnorm(1,mean = 0,sd = sd)
```

adds a randomly distributed value $\epsilon \sim N(0, \sigma^2)$ to the imputed value, where $\sigma^2 = se(\hat{\beta})$ from the original, complete-case analysis. Note also that you need to set the "seed" for the randomization by `set.seed` command. If you do not, the code above will still work, but you will be generating a different stream of random numbers each time[1]. In other words, we are doing the following:

$$\hat{y}_{ij} = \hat{\beta}_0 + x\hat{\beta}_1 + \epsilon_{ij}$$

with $i = 1, \cdots 10$ and $j = 1, \cdots, N = 1000$. The results are as follows. The estimated median and mean of beta are

```
# The estimate of beta is the sample mean or median
apply(sample.beta,2,mean)

##          it    betaHat0    betaHat1
## 500.5000000   6.5216592  -0.3519784

apply(sample.beta,2,median)

##          it    betaHat0    betaHat1
## 500.5000000   6.4886998  -0.3437068

# The standard error is the sample SD
apply(sample.beta,2,sd)

##          it    betaHat0    betaHat1
## 288.8194361   0.8809348   0.2928874
```

The histogram of the generated betas is shown in the following figure.

[1]The software sets the seed borrowing a value, usually from some internal function of the computer such as the system clock.

```r
# Histogram of betahat
hist(sample.beta[,3],30,main = "", xlab = expression(beta))
```

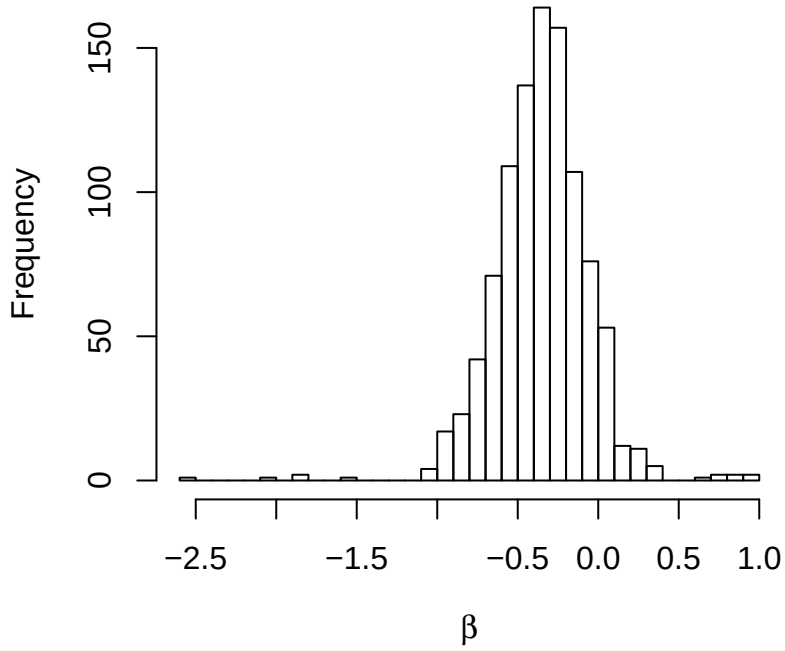

Figure 1: Histogram of simulated regression coefficient of the slope

1.2 The OASIS study

To see the implications of data missing at random, we consider the following data set. We import the data into R and the number of missing values.

```r
ESR = read.csv("h:/teaching/Indiana/PBHL B-582/labs/lab 7 (data analysis I)/data/ESR.csv")
head(ESR)
```

```
##   group status  phat_MAR statmis phat_1
## 1     A      0 0.7611941       0   0.89
## 2     A      0 0.7611941       0   0.89
## 3     A      0 0.7611941       0   0.89
## 4     A      0 0.7611941       0   0.89
## 5     A      0 0.7611941       0   0.89
## 6     A      0 0.7611941       0   0.89
```

```r
# Create labels
```

```
ESR$trt = NA
ESR$trt[ESR$group == "A"] = "ET"
ESR$trt[ESR$group == "B"] = "ST"
ESR$trt = factor(ESR$trt,levels = c("ET","ST"))

# See the number of missing values
apply(ESR,2, function(x) sum(is.na(x)) )

##    group   status phat_MAR  statmis   phat_1      trt
##        0      142        0        0        0        0
```

Before we proceed with the analysis, consider the code fragment

```
# Create labels
ESR$trt = NA
ESR$trt[ESR$group == "A"] = "ET"
ESR$trt[ESR$group == "B"] = "ST"
ESR$trt = factor(ESR$trt,levels = c("ET","ST"))
```

This code creates a new variable `trt` (at first populated by missing values), and then assigns the label `ET` in the observations from group A ("enhanced" treatment) and `ST` in the observations from group B ("standard treatment"). It then recasts this variable as a factor variable (similar to the SAS `class` variable).

1.2.1 Analysis under MCAR

First, we carry out an analysis based on the subjects with available data

```
# Analysis of completers (assuming MCAR)
x = table(ESR$status,ESR$trt)
x

##
##     ET ST
##   0 16 11
##   1 51 78

# Proportions
prop.table(x,2)

##
##            ET        ST
##   0 0.2388060 0.1235955
##   1 0.7611940 0.8764045

# Fitting a logistic regression model on the complete data
fit.CC = glm(status ~ trt,data = ESR,family = "binomial")
summary(fit.CC)
```

```
##
## Call:
## glm(formula = status ~ trt, family = "binomial", data = ESR)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0449   0.5137   0.5137   0.7387   0.7387
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.1592     0.2865   4.046 5.22e-05 ***
## trtST         0.7996     0.4311   1.855   0.0636 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 143.75  on 155  degrees of freedom
## Residual deviance: 140.24  on 154  degrees of freedom
##   (142 observations deleted due to missingness)
## AIC: 144.24
##
## Number of Fisher Scoring iterations: 4

# CI for the odds ratio
exp(confint(fit.CC))

## Waiting for profiling to be done...

##                 2.5 %   97.5 %
## (Intercept) 1.8612751 5.769456
## trtST       0.9640272 5.301382
```

If we perform a logistic-regression analysis, based on the subjects with known smoking status (completers), the smoking rate at one year is $p_{ET} = 0.76$ and $p_{ST} = 0.88$ for the enhanced and standard intervention groups respectively. The odds ratio is $\hat{\psi} = \exp(0.7996) = 2.225$ (95% confidence interval (0.96, 5.30)). Note that, from the logistic-regression output above, this is significant at the 10% but not the 5% level.

1.2.2 Analysis under MAR

We make a single imputation assuming that that the missing smoking status is missing randomly within each intervention group (MAR)

```
# Indices of missing values
sub = ESR$statmis == 1
```

```r
set.seed(1)
ESR[sub,]$status = rbinom(n = sum(sub),size = 1,
                          prob = ESR$phat_MAR[sub])

# Now we have a complete dataset
x = table(ESR$status,ESR$trt)
x
```

```
##
##      ET  ST
##   0  38  17
##   1 111 132
```

```r
# Proportions under MAR
prop.table(x,2)
```

```
##
##            ET        ST
##   0 0.2550336 0.1140940
##   1 0.7449664 0.8859060
```

```r
# Fit a logistic regression model
fit = glm(status ~ trt,data = ESR,family = "binomial")
summary(fit)
```

```
##
## Call:
## glm(formula = status ~ trt, family = "binomial", data = ESR)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0836   0.4922   0.4922   0.7673   0.7673
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.0719     0.1879   5.703 1.17e-08 ***
## trtST         0.9776     0.3189   3.065  0.00217 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 285.03  on 297  degrees of freedom
## Residual deviance: 274.99  on 296  degrees of freedom
## AIC: 278.99
##
## Number of Fisher Scoring iterations: 4
```

```r
# CI for the odds ratio
exp(confint(fit))

## Waiting for profiling to be done...

##                 2.5 %   97.5 %
## (Intercept) 2.041271 4.274097
## trtST       1.443359 5.072826
```

Under MAR, we assume that the proportions of smokers among patients who dropped out in the two groups is identical to those who did not. From this analysis, we see that the estimated odds ratio is $\hat{\psi} = \exp(0.9776) = 2.658$ (95% confidence interval (1.44, 5.07)), which is statistically significant at the 5% level.

The previous analysis has a significant drawback. It assumes that our guess about the smoking status among dropouts is exactly known. Thus, the estimate of the variability of the odds ratio will be underestimated.

To address this, we combine multiple imputation with resampling methods. First we define a new function `ERboot`.

```r
ESRboot = function(data,N = 1000, MAR = T, pmisET = NULL, pmisST = NULL)
 {
   # Data frame including the results of each iteration
   results = data.frame(it = 1:N, pST = NA, pET = NA, OR = NA)

   # Number of observations
   n = nrow(data)

   for (i in 1:N)
   {
     # Bootstrap sample
     temp_data = data[sample(n,replace = T),]

     if (MAR==T)
     {
       x = table(temp_data$status,temp_data$trt)
       prop.table(x,2)
       pmisET = prop.table(x,2)[2,"ET"]
       pmisST = prop.table(x,2)[2,"ST"]
     }

     # Indices of missing values
    subET = temp_data$statmis == 1 & temp_data$trt == "ET"
    subST = temp_data$statmis == 1 & temp_data$trt == "ST"

    if (any(subET | subST))
       {
```

```
        # Impute missing values
        temp_data[subET,]$status = rbinom(n = sum(subET),
                                          size = 1,prob = pmisET)
        temp_data[subST,]$status = rbinom(n = sum(subST),
                                          size = 1,prob = pmisST)
        }

    x = table(temp_data$status,temp_data$trt)
    x

    # Proportions under MAR with imputation
    prop.table(x,2)

    results[i,"OR"] = x[1,1]*x[2,2]/(x[2,1]*x[1,2])
    results[i,"pET"] = prop.table(x,2)[2,"ET"]
    results[i,"pST"] = prop.table(x,2)[2,"ST"]
   }
   return(results)
}
sampleOR = ESRboot(ESR)
```

The analysis is as follows:

```
# The estimates are the sample means or medians
apply(sampleOR,2,mean)

##          it         pST         pET          OR
## 500.5000000   0.8790533   0.7509644   2.7529571

# The standard error is the sample SD
apply(sampleOR,2,sd)

##           it          pST          pET           OR
## 288.81943610   0.03489803   0.04700226   1.50385074
```

The histogram of the imputed odds ratios is shown in Figure 2.

1.2.3 Analysis under MNAR

In most cases, the subjects of a study who stay on observation are not representative of those who drop out in ways that are not predictable. We say that the data are missing not at random (MNAR). In the OASIS study, four experts were consulted and asked to provide their expert opinion about the likelihood that the subjects who discontinued from the study continue to abstain from smoking.

```
hist(sampleOR$OR,50,main = "", xlab = "Odds ratio")
```

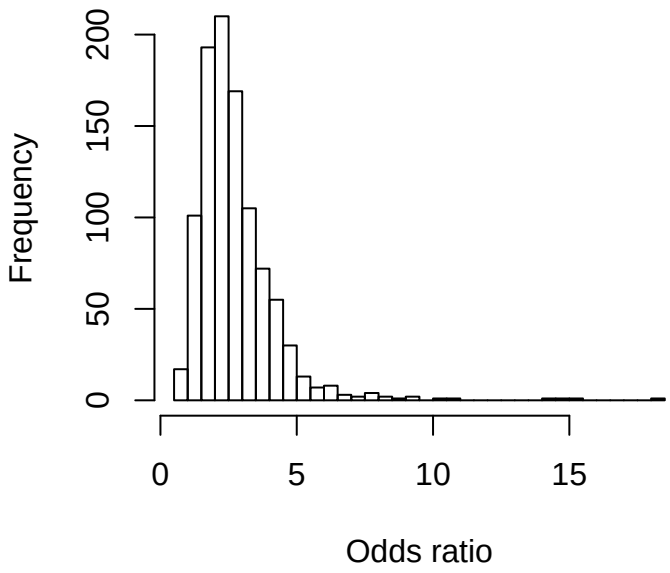

Figure 2: Histogram of the odds ratio

We undertake the anayses based on their opinion (the "scenarios" listed below).

```
scenarios = matrix(nr = 2,nc = 4)
rownames(scenarios) = c("ET","ST")
scenarios[1,] = c(0.83,0.87,0.87,0.83)
scenarios[2,] = c(0.90,0.91,0.90,0.90)

sampleOR = ESRboot(ESR, MAR = F, pmisET = 0.7611941, pmisST = .8764045)
```

The estimated odds ratios, resulting from the (rather optimistic) analysis under MAR and those derived from the opinion of the four experts, is show in Figure 3.

```r
# Analysis under MAR
plot(density(sampleOR$OR),xlab = "Odds ratio",
     xlim = c(0,6),col = "green", main="", lwd = 2,ylim = c(0,1.0))

# Scenarios according to the four experts
for (j in 1:ncol(scenarios))
 {
   sampleOR = ESRboot(ESR, MAR = F, pmisET = scenarios[1,j],
                      pmisST = scenarios[1,j])
   lines(density(sampleOR$OR),lwd = 2,lty = j)
}
```

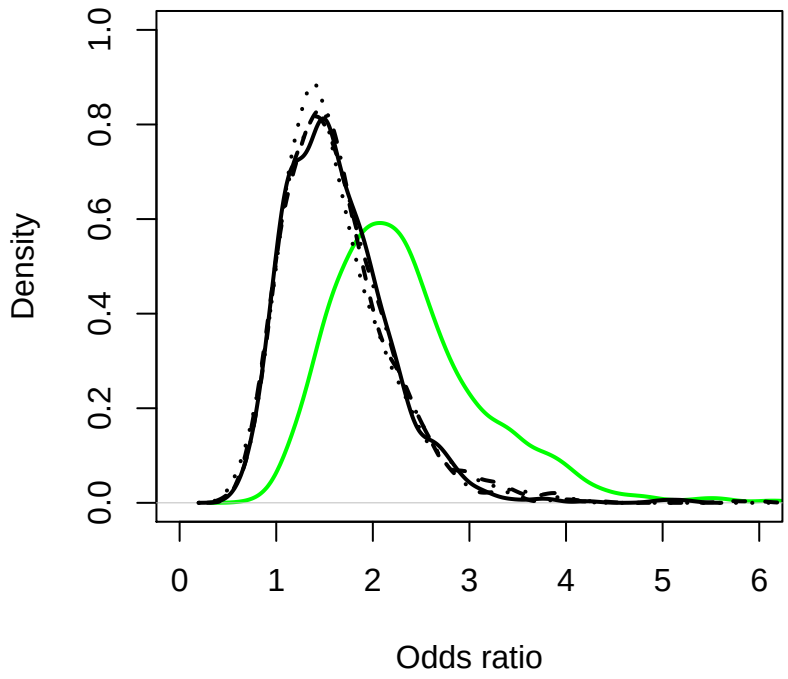

Figure 3: Monte Carlo simulations under MAR (green) and the four experts .