Incomplete Data Analysis, Assignment 1, Exercise 1+2

Miriam Fischer

17 10 2020

Exercise 1

1a: Correct answer: ii, 0.3

The correct answer is ii. If ALQ is missing completely at random, the missingness of ALQ must be independent of the value of ALQ itself. That means, whether ALQ is missing does not depend on whether ALQ is (would have been) Yes or No. Therefore, the probability for missingness given the variable is Yes must be the same as it is for given it is No. As the probability of ALQ missing given ALQ=Yes has 0.3, so must e the probability of missing given ALQ=No.

1b: Correct answer: ii, P(ALQ missing) is independent of Yes No value of ALQ after adjusting for gender

The correct answer is ii. Missing at random means that the missingness is allowed to depend on other variables (in our case gender). However, if we stratify on gender, the probability of missingness then must be independent of ALQ, given gender.

1c: Correct answer: iii, We do not have enough information

The correct answer is iii. We know that $P(ALQ = missing \mid men) = 0.1$. Further we only have one variable which we condition on, with only two possible events (which are exclusive). The law of total probability gives us $P(A) = \sum_n P(A \mid B_n) * P(B_n)$.

Here, it corresponds to $P(ALQ = missing) = P(ALQ = missing \mid Male) * P(Male) + P(ALQ = missing \mid Female) * P(Female)$. In order to calculate $P(ALQ = missing \mid Female)$ we must know the proportions of male to females in our dataset, and the total probability of missing values. However, these information is not given. So we do not have enough information to answer the question (however, the information we need we can find in the data).

Exercise 2: Correct answer Biggest: N' = 90 , smallest: N' = 0

We have $N = 100$ and $x_1, \ldots, x_{10}$ variables. Each of the ten variables x_i contains $100 * 0.1 = 10$ missing values. Under a complete case analysis, we only consider observations which have data in all of the ten variables. The largest sample we can get is if all ten variables have their missing values in the same observation. We know that each variable has 10 missing values, so if all variables have their missing values at same observations, there are 10 observations with missing data (in this case, they would not have any data). Therefore, the largest possible subsample is $100 - 10 = 90$.

Regarding the smallest possible subset, all variables have their missing values at different observations. For an observation to not be included in the complete case analysis, it is already enough if only one variable has a missing value. For simplicity, we assume the missingness in the variables is ordered, e.g. variable x_1 has missing values for observations 1 to 10, variable x_2 has missing values for observations 11 to 20, and so forth. We can then see that the the smallest possible subset is actually 0, that means if we are unlucky, the complete case analysis would remove all our data!

Information to Git-repository

The git-repository for Exercise 3,4 can be found at the following URL:

https://github.com/miribella/MF_missing_data_assignment1.git

Exercise3

Miriam Fischer

12 10 2020

Creating Dataset

We first create our random variables, and the data of Y1 and Y2:

```r
set.seed(40)
Z1 <- rnorm(500, 0, 1)
Z2 <- rnorm(500, 0, 1)
Z3 <- rnorm(500, 0, 1)

Y1 <- 1 + Z1
Y2 <- 5 + 2 * Z1 + Z2
```

We then set the function when Y2 should be missing:

```r
a <- 2
b <- 0
v <- a * (Y1 - 1) + b * (Y2 - 5) + Z3
```

3a: Impose missingness on Y2

We impose missingness on Y2 for Exercise 3a

```r
ind_a <- which(v < 0)
Y2_MAR_obs <- Y2[-ind_a]
Y2_MAR_mis <- Y2[ind_a]
```

Note that this is MAR, missing at random. The reason is that with a=2 and b=0, the function which determines missingness on Y2 is

$$2(Y_1 - 1) + Z_3 < 0$$

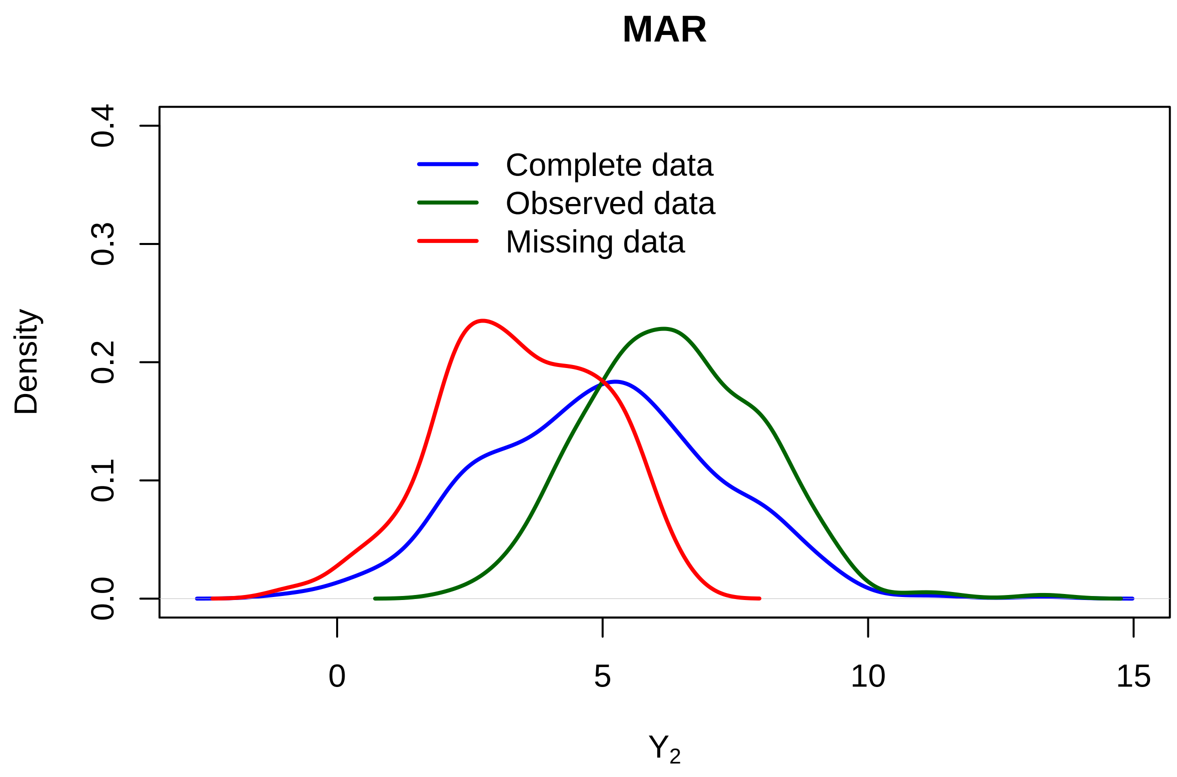
This function clearly depends on Y1, but not on Y2. It also depends on a random variable Z3 (which is without connection to Y1 and Y2). We note that Y1 is fully observed. Thus, missingness of Y2 depends on the value of Y1, which we have available in our dataset, making it MAR. Most importantly, missingness does not depend on Y2 itself. We further argue that the missingness depending on Z3 is not seen as missingness depending on an un-observed variable (in which case we would have MNAR), as Z3 is solely random and thus the missingness conditioned on Z3 is random.

3a: Plots

We now plot the missing data for 3a.

```r
plot(density(Y2), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "MAR", ylim = c(0,
    0.4))
lines(density(Y2_MAR_obs), lwd = 2, col = "darkgreen")
lines(density(Y2_MAR_mis), lwd = 2, col = "red")
```

```r
legend(1, 0.4, legend = c("Complete data", "Observed data", "Missing data"), col = c("blue",
    "darkgreen", "red"), lty = c(1, 1, 1), lwd = c(2, 2, 2), bty = "n")
```



As we can see, the plots of the observed and missing data are not the same. However, they are shifted: if we ignore the fact that their respective mean and mode are at different values of Y2, the shape of the plot is similar. With that I mean that if we would shift the curve of the missing data to the right, we would roughly get the curve of the observed data. This is commonly seen for MAR data: Unconditioned on observed variables, missingness looks MNAR, but if we condition on observed variables, the missingness follows the same patterns.

3b: Regression imputation

We perform linear regression and assume that the necessary requirements for linear regression are fulfilled. We want to estimate the effect of Y1 on Y2.

```r
Y2_MAR_na <- ifelse(v < 0, NA, Y2)
data_b <- data.frame(Y1_reg_b = Y1, Y2_reg_b = Y2_MAR_na)
reg_fit_b <- lm(Y2_reg_b ~ Y1_reg_b, data <- data_b)
```

The regression is:

```r
reg_fit_b$coefficients
```

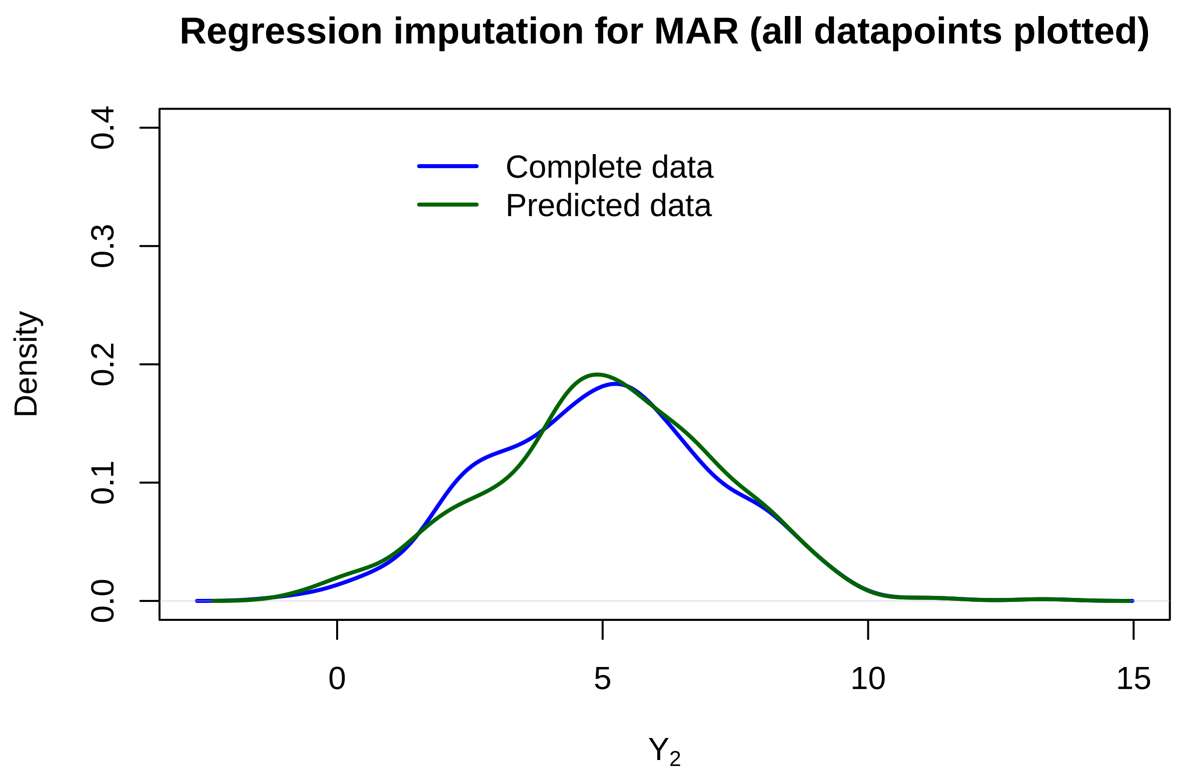
```
## (Intercept)    Y1_reg_b
##    2.989912    1.942356
```

Our predicted dataset will have the known values of Y2 for the values where we did not impose missingness, and will use the values predicted with our regression for the values which are missing. Further, as we use stochastic imputation, we include a random noise.

```
predicted_b <- predict(reg_fit_b, newdata = data_b) + rnorm(nrow(data_b), 0, sigma(reg_fit_b))
Y2_MAR_pre <- ifelse(is.na(data_b$Y2_reg_b), predicted_b, Y2)
```

Plots

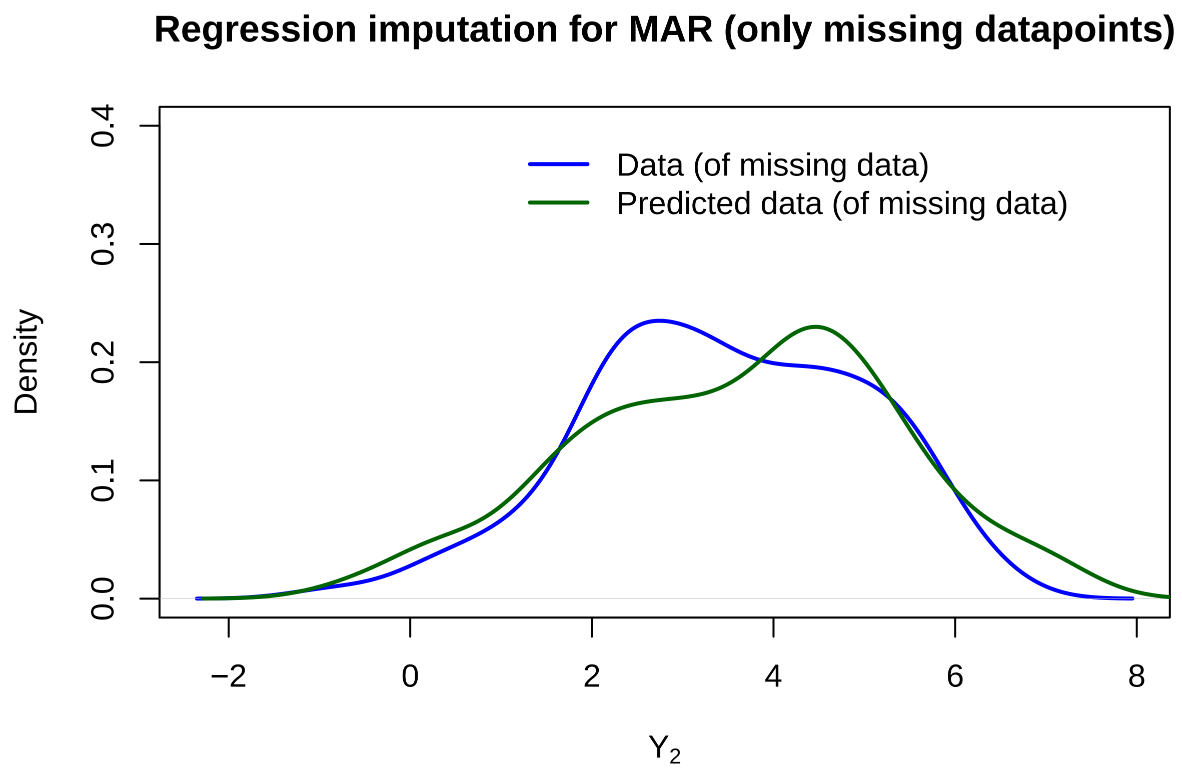
```
# plot
plot.new()
frame()
plot(density(Y2), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "Regression imputation for MAR
    ylim = c(0, 0.4))
lines(density(Y2_MAR_pre), lwd = 2, col = "darkgreen")
legend(1, 0.4, legend = c("Complete data", "Predicted data"), col = c("blue", "darkgreen"),
    lty = c(1, 1), lwd = c(2, 2), bty = "n")
```



We note that for datapoints where Y2 is not missing, the “predicted” value and the correct value are equal (as, looking at our code, we our predicted data does not predict the values for given datapoints, but just takes their known values). We therefore use a second graph, which only has the data which we actually predict.

```
plot.new()
frame()
Y2_MAR_onlypre <- Y2_MAR_pre[ind_a]
plot(density(Y2_MAR_mis), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "Regression imputation
    ylim = c(0, 0.4))
lines(density(Y2_MAR_onlypre), lwd = 2, col = "darkgreen")
```

```r
legend(1, 0.4, legend = c("Data (of missing data)", "Predicted data (of missing data)"),
    col = c("blue", "darkgreen"), lty = c(1, 1), lwd = c(2, 2), bty = "n")
```



3c: Impose missingness

We change the parameters when Y2 should be missing:

```r
a <- 0
b <- 2
vv <- a * (Y1 - 1) + b * (Y2 - 5) + Z3
```

We impose missingness on Y2 for Exercise 3c

```r
ind_c <- which(vv < 0)
Y2_MNAR_obs <- Y2[-ind_c]
Y2_MNAR_mis <- Y2[ind_c]
```

Note that this is MNAR, missing not at random. The reason is that with a=0 and b=2, the function which determines missingness on Y2 is

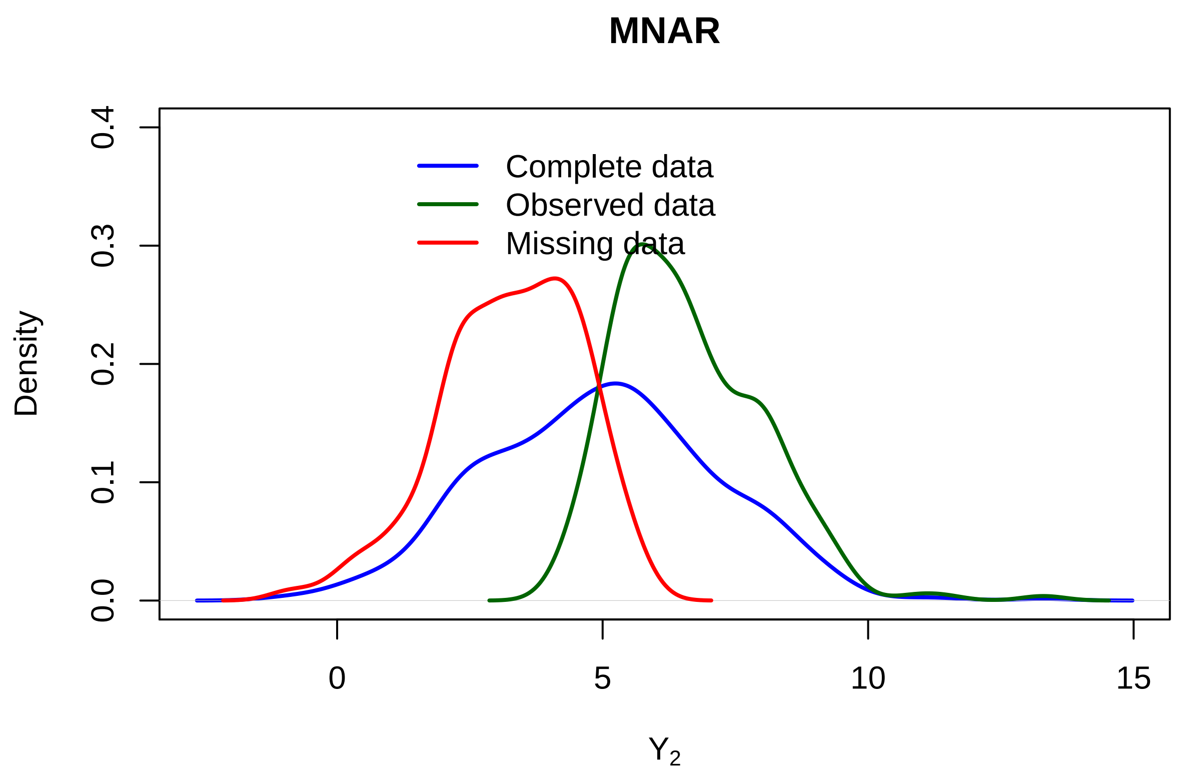
$$2(Y_2 - 5) + Z_3 < 0$$

This function clearly depends on Y2, but not on Y1. Thus, must crucially, missingness of Y2 depends on the value of Y2 itself, making it MNAR.

3c: Plots

We now plot the missing data for 3c.

```r
plot(density(Y2), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "MNAR",
    ylim = c(0, 0.4))
lines(density(Y2_MNAR_obs), lwd = 2, col = "darkgreen")
lines(density(Y2_MNAR_mis), lwd = 2, col = "red")
legend(1, 0.4, legend = c("Complete data", "Observed data", "Missing data"), col = c("blue",
    "darkgreen", "red"), lty = c(1, 1, 1), lwd = c(2, 2, 2), bty = "n")
```



Most importantly, the missing data and the observed data are very different to each other. This will prove to be different to predict the missing data later.

3d: Regression imputation

We again use linear regression and assume the underlying assumptions to be met.

```r
Y2_MNAR_na <- ifelse(vv < 0, NA, Y2)
data_d <- data.frame(Y1_reg_d = Y1, Y2_reg_d = Y2_MNAR_na)
reg_fit_d <- lm(Y2_reg_d ~ Y1_reg_d, data <- data_d)
```

The regression is:

```r
reg_fit_d$coefficients
```

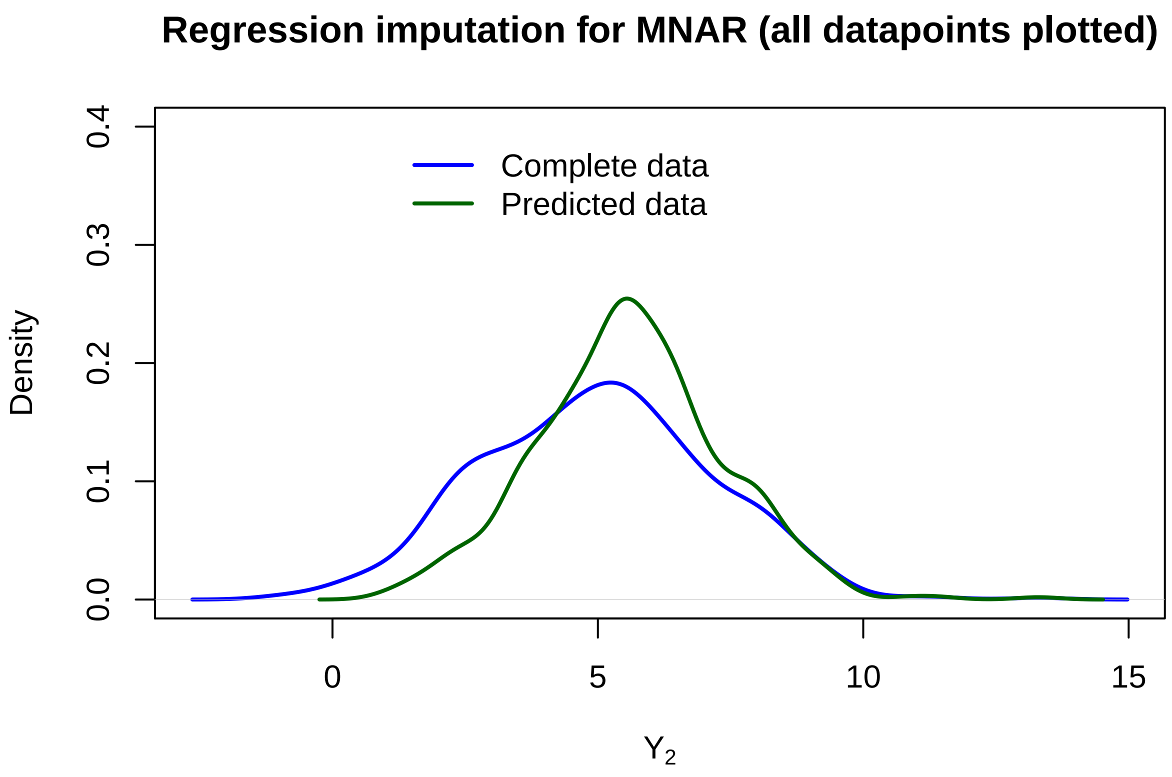
```
## (Intercept)    Y1_reg_d
##    4.018839    1.500059
```

Our predicted dataset will have the known values of Y2 for the values where we did not impose missingness, and will use the values predicted with our regression for the values which are missing.

```
predicted_d <- predict(reg_fit_d, newdata = data_d) + rnorm(nrow(data_d), 0, sigma(reg_fit_d))
Y2_MNAR_pre <- ifelse(is.na(data_d$Y2_reg_d), predicted_d, Y2)
```

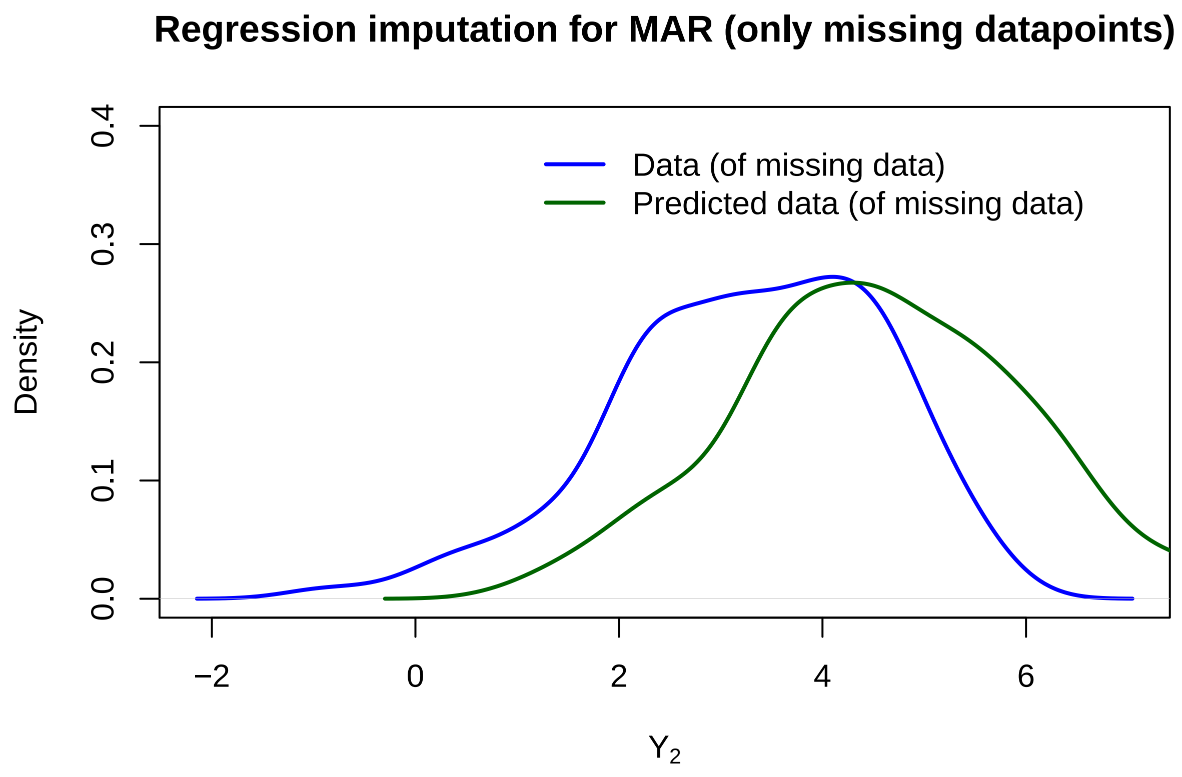
Plots

```
# plot
plot.new()
frame()
plot(density(Y2), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "Regression imputation for MNAI
    ylim = c(0, 0.4))
lines(density(Y2_MNAR_pre), lwd = 2, col = "darkgreen")
legend(1, 0.4, legend = c("Complete data", "Predicted data"), col = c("blue", "darkgreen"),
    lty = c(1, 1), lwd = c(2, 2), bty = "n")
```



We note that for datapoints where Y2 is not missing, the "predicted" value and the correct value are equal (as, looking at our code, we our predicted data does not predict the values for given datapoints, but just takes their known values). We therefore use a second graph, which only has the data which we actually predict.

```
plot.new()
frame()
Y2_MNAR_onlypre <- Y2_MNAR_pre[ind_c]
plot(density(Y2_MNAR_mis), lwd = 2, col = "blue", xlab = expression(Y[2]), main = "Regression imputatio
    ylim = c(0, 0.4))
lines(density(Y2_MNAR_onlypre), lwd = 2, col = "darkgreen")
legend(1, 0.4, legend = c("Data (of missing data)", "Predicted data (of missing data)"),
    col = c("blue", "darkgreen"), lty = c(1, 1), lwd = c(2, 2), bty = "n")
```



Compared to the plots of the prediction for MAR, the predictions we are doing here are much worse: for MAR, our predictions were somewhat accurate (or at least reasonable), but the predictions we have here are far from being accurate. Thus, for MNAR, our prediction gives very poor results in predicting Y2. This is not a surprise: Our regression tries to predict Y2 based on Y1, but for our model, missingness on Y2 is not based on Y1, but on Y2. It is no surprise that the prediction is not good, as it tries to find a causality between two variables which do not have a causal connection to each other! More badly, for the regression the noise we add is probably doing better in predicting Y2 than Y1 itself (I haven't checked this though, but I find it likely to be the case)!

Exercise 4

Miriam Fischer

12 10 2020

We load our data (please use your corresponding path)

```
load("C:/Users/Miriam/Box Sync/Masters/IncompleteDataAnalysis/Assignment1_MF/databp.Rdata")
```

4a: Complete case analysis

We create a dataset only of complete observations

```
data_complete <- databp[complete.cases(databp), ]
```

We calculate the mean and standard error of the complete data For the standard error, we define a function calculating the error.

```
mean(data_complete$recovtime)
```

```
## [1] 19.27273
```

```
# sterror = st deviation divided by squareroot of length of data
sterror <- function(x) sd(x)/sqrt(length(x))
sterror(data_complete$recovtime)
```

```
## [1] 2.603013
```

To calculate the pearson correlation, we need to first calculate the correlation matrix (excluding missing cases). We then divide the corresponding entry by the product of the two variables standard deviation. We note than when calculating their standard deviation, we must use only the entries included in our complete case analysis.

```
covcc <- cov(databp, use = "complete")
```

correlation between recovery and dose

```
covcc["logdose", "recovtime"]/(sd(data_complete$logdose) * sd(data_complete$recovtime))
```

```
## [1] 0.2391256
```

correlation between time and bpressure

```
covcc["bloodp", "recovtime"]/(sd(data_complete$bloodp) * sd(data_complete$recovtime))
```

```
## [1] -0.01952862
```

4b: Mean imputation

We first observe that the time is the only variable with missing observations, we thus only need to perform imputation on this variable

```
mean_recov <- mean(databp$recovtime, na.rm = TRUE)
recov_mi <- ifelse(is.na(databp$recovtime), mean_recov, databp$recovtime)
```

Mean and st error of recovtime with mean imputation

```
data_mi <- data.frame(logdose = databp$logdose, bloodp = databp$bloodp, recovtime = recov_mi)

mean(data_mi$recovtime)
```

```
## [1] 19.27273
```

```
sterror(data_mi$recovtime)
```

```
## [1] 2.284135
```

We again calculate the pearson correlation, this time on our new dataset with the missing values filled in with the mean (and we don't need to say we only want complete observations as the data is complete)

```
covmi <- cov(data_mi)
```

correlation between recovery and dose

```
covmi["logdose", "recovtime"]/(sd(data_mi$logdose) * sd(data_mi$recovtime))
```

```
## [1] 0.2150612
```

correlation between time and bpressure

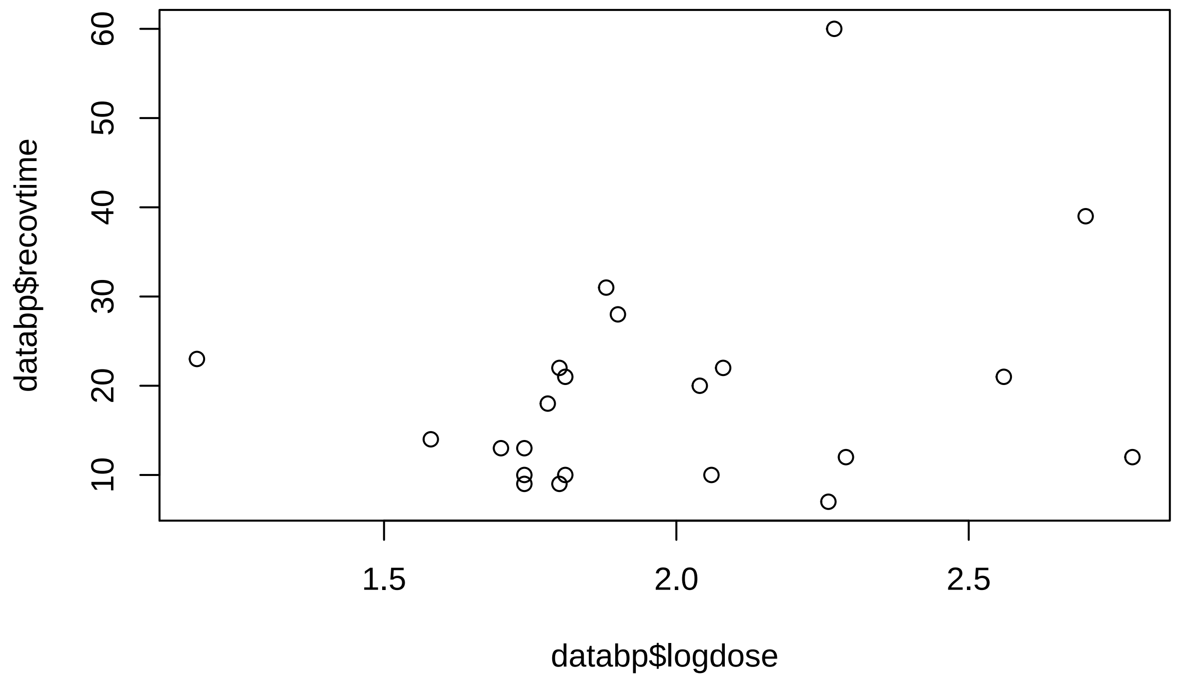
```
covmi["bloodp", "recovtime"]/(sd(data_mi$bloodp) * sd(data_mi$recovtime))
```

```
## [1] -0.01934126
```

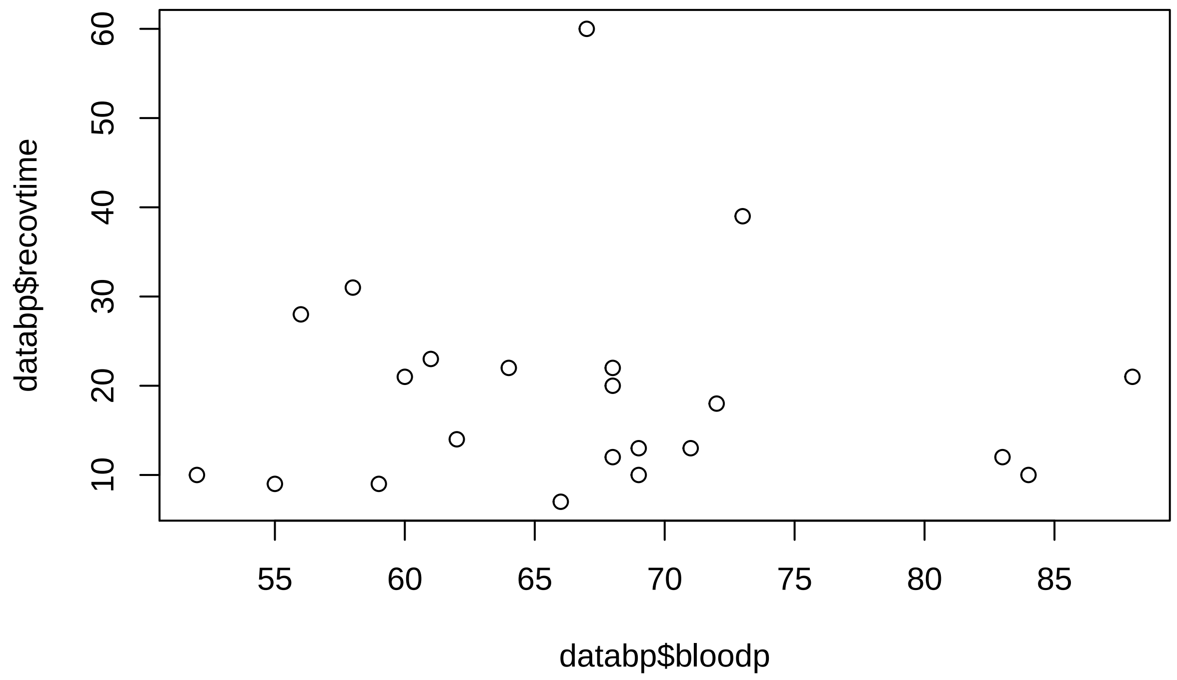
4c: Mean regression imputation

We first plot the variables, to see if the assumption of a linear dependence used for our regression is accurate. Whereas I wouldn't be really confident saying there is a linear dependence, the plots give at least no evidence to say the dependence would be for example logarithmic or quadratic. So in my opinion, this data is more telling us there is no reason not to do linear regression, more than they say there is reason to do linear regression.

```
plot(databp$logdose, databp$recovtime)
```



```r
plot(databp$bloodp, databp$recovtime)
```



We include both of our observed variables into our regression. Our regressionmodel is

$$\text{Recovtime}_i = \beta_0 + \beta_1 * \text{Logdose}_i + \beta_2 * \text{Bloodpressure}_i + \varepsilon_i$$

assuming $\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$. We expect $\beta_1 > 0$ and $\beta_2 < 0$. As we have seen a positive correlation between time and dose and a negative correlation between time and bpressure.

```
fit_mr <- lm(databp$recovtime ~ databp$logdose + databp$bloodp)
fit_mr$coefficients
```

```
##    (Intercept) databp$logdose  databp$bloodp
##     15.2159065     11.4290287     -0.2769265
```

We create a new vector with the predicted times and include it to a new dataset with our new predicted values.

```
predicted_time <- predict(fit_mr, newdata = databp)
recov_predict <- ifelse(is.na(databp$recovtime), predicted_time, databp$recovtime)
data_mr <- data.frame(logdose = databp$logdose, bloodp = databp$bloodp, recovtime = recov_predict)
```

Mean and standard error of the predicted dataset

```
mean(data_mr$recovtime)
```

```
## [1] 19.44428
```

```
sterror(data_mr$recovtime)
```

```
## [1] 2.312845
```

Calculating the correlation matrix

```
covmr <- cov(data_mr)
```

correlation between recovery and dose

```
covmr["logdose", "recovtime"]/(sd(data_mr$logdose) * sd(data_mr$recovtime))
```

```
## [1] 0.2801835
```

correlation between time and bpressure

```
covmr["bloodp", "recovtime"]/(sd(data_mr$bloodp) * sd(data_mr$recovtime))
```

```
## [1] -0.0111364
```

4d:

We use the model of 3c and add a random number

```
set.seed(49)

predicted_sri <- predict(fit_mr, newdata = databp) + rnorm(nrow(databp), 0, sigma(fit_mr))
recov_sri <- ifelse(is.na(databp$recovtime), predicted_sri, databp$recovtime)
```

We calculate the mean and standard error

```
data_sri <- data.frame(logdose = databp$logdose, bloodp = databp$bloodp, recovtime = recov_sri)
mean(data_sri$recovtime)
```

```
## [1] 18.72278
```

```
sterror(data_sri$recovtime)
```

```
## [1] 2.341176
```

Calculating the correlation matrix

```
covsri <- cov(data_sri)
```

correlation between recovery and dose

```
covsri["logdose", "recovtime"]/(sd(data_sri$logdose) * sd(data_sri$recovtime))
```

```
## [1] 0.251039
```

correlation between time and bpressure

```
covsri["bloodp", "recovtime"]/(sd(data_sri$bloodp) * sd(data_sri$recovtime))
```

```
## [1] -0.01514032
```

As we include a random error term, we need to make sure that we don't predict negative times. Further, we might consider to limit the fractional part of the predicted values, thus we might include a floor function to our predicted values, making them nonfractional. In that case, we would use

```
recov_sri_int <- ifelse(is.na(databp$recovtime), floor(predicted_sri), databp$recovtime)
data_sri_int <- data.frame(logdose = databp$logdose, bloodp = databp$bloodp, recovtime = recov_sri_int)
```

giving us the following mean, standard error, correlation matrix:

```
mean(data_sri_int$recovtime)
```

```
## [1] 18.64
```

```
sterror(data_sri_int$recovtime)
```

```
## [1] 2.347964
```

```
covsriint <- cov(data_sri_int)
covsriint["logdose", "recovtime"]/(sd(data_sri_int$logdose) * sd(data_sri_int$recovtime))
```

```
## [1] 0.2479823
```

```
covsriint["bloodp", "recovtime"]/(sd(data_sri_int$bloodp) * sd(data_sri_int$recovtime))
```

```
## [1] -0.01684754
```

4e: predictive mean

We define a function which will calculate the index of the closest observed value to our prediction.

```
mis_index <- which(is.na(databp$recovtime))

best_replace <- function(val) {

    min <- .Machine$integer.max
    ind <- -1
    available_data <- which(is.na(databp$recovtime) == FALSE)

    for (i in available_data) {
        squaredis <- (val - databp$recovtime[i])^2
        if (squaredis < min) {
            min <- squaredis
            ind <- i
        }
    }
    return(ind)
}

recov_prmtp <- 1:length(databp$recovtime)

for (inde in mis_index) {
    best <- best_replace(predicted_sri[inde])
    recov_prmtp[inde] <- databp$recovtime[best]
    cat("datapoint ", inde, " we take value ", databp$recovtime[best], " of index ",
        best, "\n")
}
```

```
## datapoint  4  we take value  7  of index  1
## datapoint  10  we take value  21  of index  7
## datapoint  22  we take value  18  of index  3
```

```
recov_prm <- ifelse(is.na(databp$recovtime), recov_prmtp, databp$recovtime)
data_prm <- data.frame(logdose = databp$logdose, bloodp = databp$bloodp, recovtime = recov_prm)
```

We calculate the mean and st error

```
mean(data_sri_int$recovtime)
```

```
## [1] 18.64
```

```
sterror(data_sri_int$recovtime)
```

```
## [1] 2.347964
```

and the correlation

```
covprm <- cov(data_prm)
covprm["logdose", "recovtime"]/(sd(data_prm$logdose) * sd(data_prm$recovtime))
```

```
## [1] 0.2590259
```

```
covprm["bloodp", "recovtime"]/(sd(data_prm$bloodp) * sd(data_prm$recovtime))
```

```
## [1] -0.01407195
```

4f:

One advantage of predictive mean matching is that we can be certain that our predicted values are values which comply with the pattern of the given variable. With that I mean that if our variable is time, and has only integer values, predictive mean matching will also only predict integer values (as it simply takes already given values). Stochastic regression imputation on the other side calculates the value with regression, and might predict non-integer values. However, predictive mean matching can have similar disadvantages to mean imputation: using already existing values as the values for missing data makes the overall data less variable (as we do not add new values to the dataset, but just have already existing data with higher frequency in the data afte the imputation), leading to changes in the standard deviation of the data.