Incomplete Data Analysis: Assignment 1

Xiao Heng (s2032451)

1.

(a) **(ii) 0.3. Since ALQ is MCAR, its probability that a value is missing is independent.** So $Pr(R=0|ALQ=No) = Pr(R=0|ALQ=Yes) = Pr(R=0) = 0.3$.

(b) **(ii) The probability of ALQ being missing is independent of the Yes/No value of ALQ after adjusting for gender.** Because the probability that MAR variable is missing only (conditionally) depends on observed informaiton (gender), but further unrelated to the missing Yes/No value of ALQ itself.

(c) **(iii) It is impossible to conclude from the information given.** Due to the conditional dependence property of MAR, it indicates that for different category of gender, the missing probability of ALQ should be significantly different, so only given missing probability for men, we cannot derive the associated missing probability for women.

2.

(i) **Largest possible complete subsample: 90.** In this case, since it is possible to have an entire line of NAs, once all missing values are in the same indices, e.g., the first 10% subsamples, then the rest part would be a complete set.

(ii) **Smallest possible complete subsample: 0.** In this case, every subject contains exactly only 1 missing value, causing the number of subjects with missing variables as large as possible.

3.

(a) First, we generate the data Y_1 and Y_2:

$$Y_1 = 1 + Z_1,$$

$$Y_2 = 5 + 2Z_1 + Z_2.$$

```r
# simulating the data
n=500
set.seed(1)
Z1 <- rnorm(n, 0, 1)
Z2 <- rnorm(n, 0, 1)
Z3 <- rnorm(n, 0, 1)
Y1 <- 1+Z1
Y2 <- 5+2*Z1+Z2
```

After having generated the data, missingness will then be imposed on Y_2, following the given formula (Y_2 is missing if):

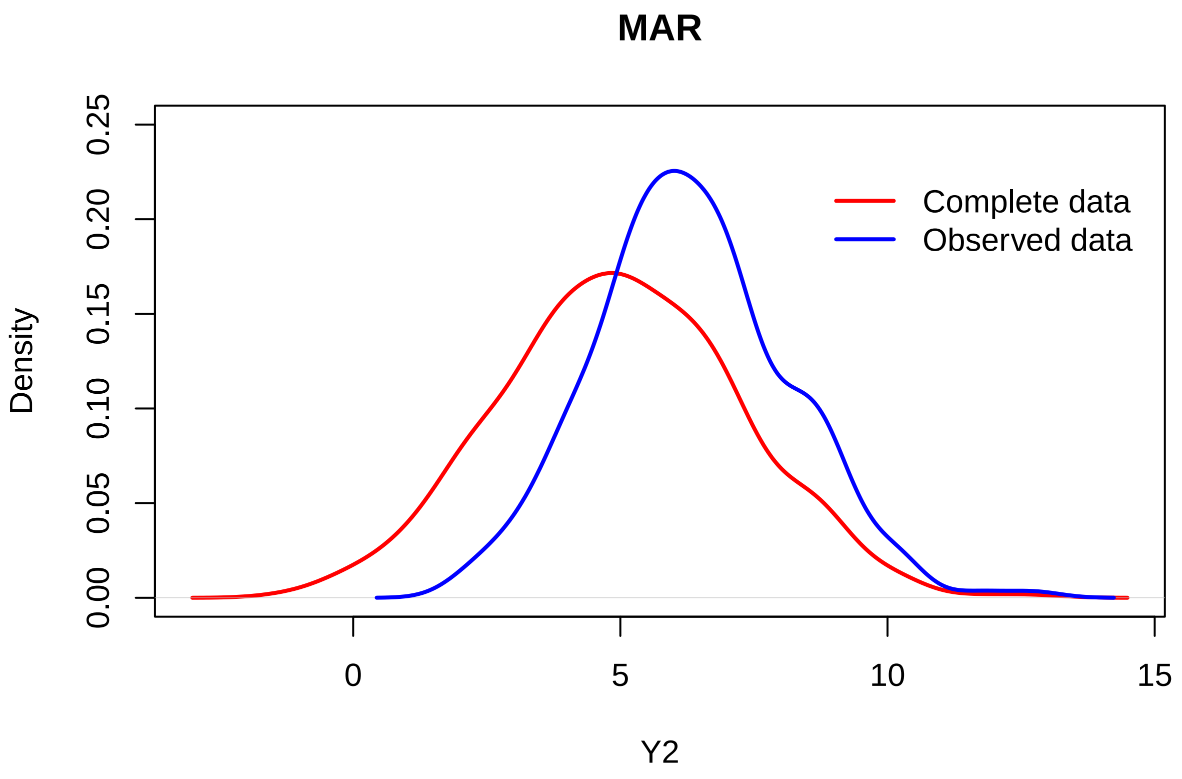
$$a \times (Y_1 - 1) + b \times (Y_2 - 5) + Z_3 < 0$$

```r
# extracting the index associated to the missing values
a_mar <- 2
b_mar <- 0
f_mar <- a_mar*(Y1-1)+b_mar*(Y2-5)+Z3
ind_mar_mis <- which(f_mar < 0)
```

```r
Y2_MAR_obs <-Y2[-ind_mar_mis]
Y2_MAR_mis <-Y2[ind_mar_mis]
```

Note that, due to the parameter $a = 2$ and $b = 0$, the missingness of Y_2 is only related to observed complete variable Y_1 and random variable Z_3, which indicates that in this case Y_2 is under MAR mechanism. Now we produce the graph as follows:

```r
plot(density(Y2),
     col = "red", ylim = c(0, 0.25), main = "MAR",
     xlab = "Y2", ylab = "Density", lwd = 2)
lines(density(Y2_MAR_obs), col = "blue", lwd = 2)
legend(8.5, 0.23, legend = c("Complete data", "Observed data"),
       col = c("red", "blue"), lty = c(1, 1), bty ="n", lwd = c(2,2))
```



We can observe that under the MAR mechanism, the two distributions are not similar with each other.

(b) In the beginning, we create a data frame for regression convenience.

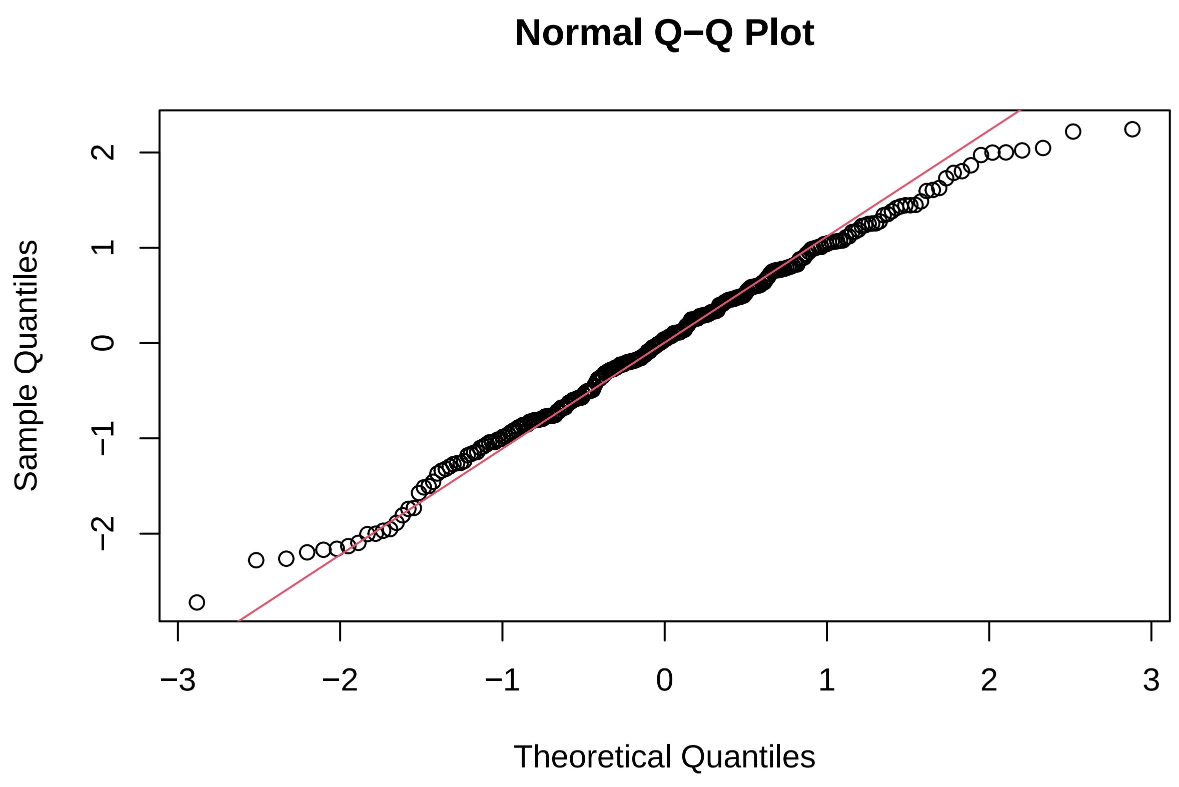
```r
# construct data.frame
Y2_MAR_na <- rep(NA,n)
ind_mar_obs <- which(f_mar >= 0)
for(i in ind_mar_obs)
  Y2_MAR_na[i] <- Y2[i]
```

For the stochastic regression imputation, we add noise to the predictions from linear regression. Since there are random numbers generated, we still set a (different) seed here.

```r
# stochastic regression imputation
data <- data.frame("y1" = Y1, "y2" = Y2_MAR_na)
fit <- lm(y2 ~ y1, data = data)
# summary(fit)
fit
```

```
##
## Call:
## lm(formula = y2 ~ y1, data = data)
##
## Coefficients:
## (Intercept)           y1
##        2.87         2.00
```

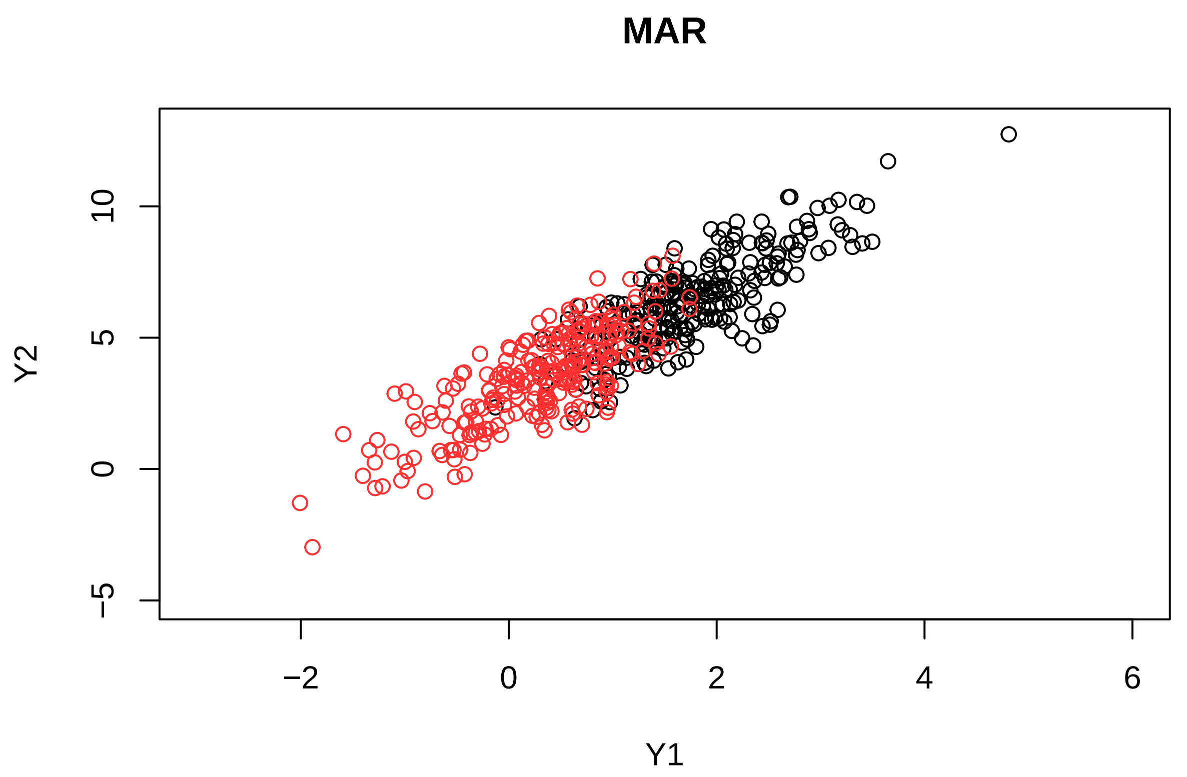
```r
qqnorm(rstandard(fit))
qqline(rstandard(fit), col = 2)
```



```r
set.seed(2)
predicted_Y2 <- predict(fit, newdata = data) + rnorm(n, 0, sigma(fit))
Y2_sri <- ifelse(is.na(Y2_MAR_na), predicted_Y2, Y2_MAR_na)
```

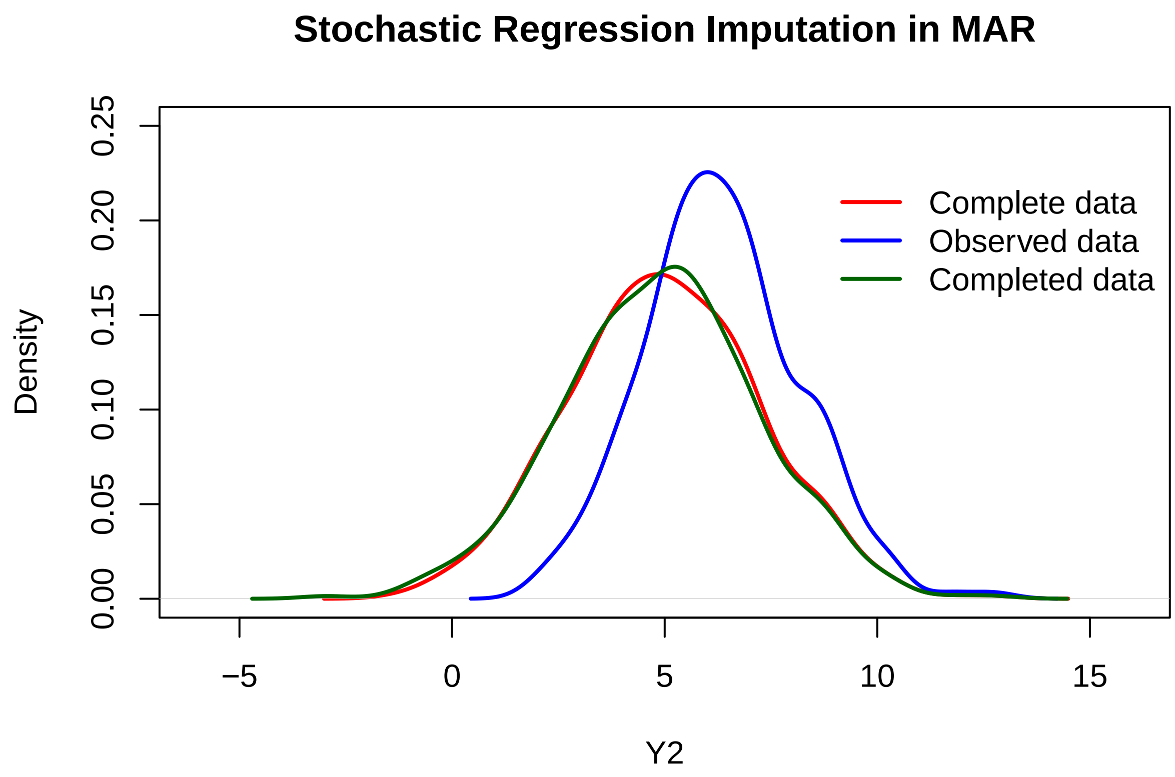
Although not required, we still show the scatter plot with the additional points in red.

```r
# scatter plot
plot(x = data$y1, y = data$y2, xlab = "Y1", ylab = "Y2", xlim = c(-3,6),
     ylim = c(-5,13), main = "MAR")
for(i in ind_mar_mis)
  points(x = Y1[i], y = Y2_sri[i], col = "firebrick1")
```



Now, with comparison of complete data and biased observed data, we find that the distribution of imputed data by stochastic regression imputation fits the original shape well, as follows:

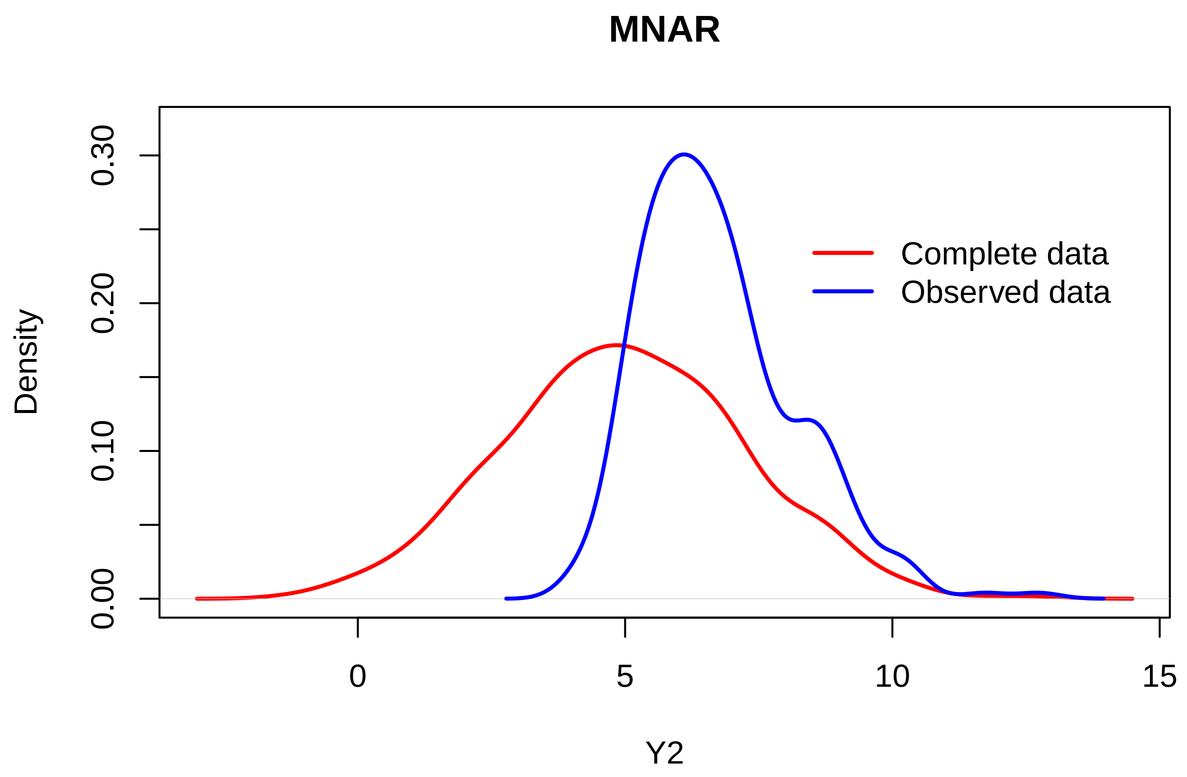
```
# distribution plot
plot(density(Y2),
     col = "red", ylim = c(0, 0.25), main = "Stochastic Regression Imputation in MAR",
     xlab = "Y2", ylab = "Density", xlim = c(-6,16), lwd = 2)
lines(density(Y2_MAR_obs), col = "blue", lwd = 2)
lines(density(Y2_sri), col = "darkgreen", lwd = 2)
legend(8.5, 0.23, legend = c("Complete data", "Observed data", "Completed data"),
       col = c("red", "blue", "darkgreen"), lty = c(1, 1, 1), bty ="n", lwd = c(2,2,2))
```



The reason is that, stochastic regression imputation gives unbiased parameter estimates (correlations, regression coefficients, etc) under a MAR missing data mechanism.

(c) With a similar process of (a), we can easily impose the missing mechanism by change the hyper-parameter value of a and b, and the corresponding distributions are shown as below.

```
# extracting the index associated to the missing values
a_mnar <- 0
b_mnar <- 2
f_mnar <- a_mnar*(Y1-1)+b_mnar*(Y2-5)+Z3
ind_mnar_mis <- which(f_mnar < 0)
Y2_MNAR_obs <-Y2[-ind_mnar_mis]
Y2_MNAR_mis <-Y2[ind_mnar_mis]
# plot
plot(density(Y2),
     col = "red", ylim = c(0, 0.32), main = "MNAR",
     xlab = "Y2", ylab = "Density", lwd = 2)
lines(density(Y2_MNAR_obs), col = "blue", lwd = 2)
legend(8, 0.26, legend = c("Complete data", "Observed data"),
       col = c("red", "blue"), lty = c(1, 1), bty ="n", lwd = c(2,2))
```



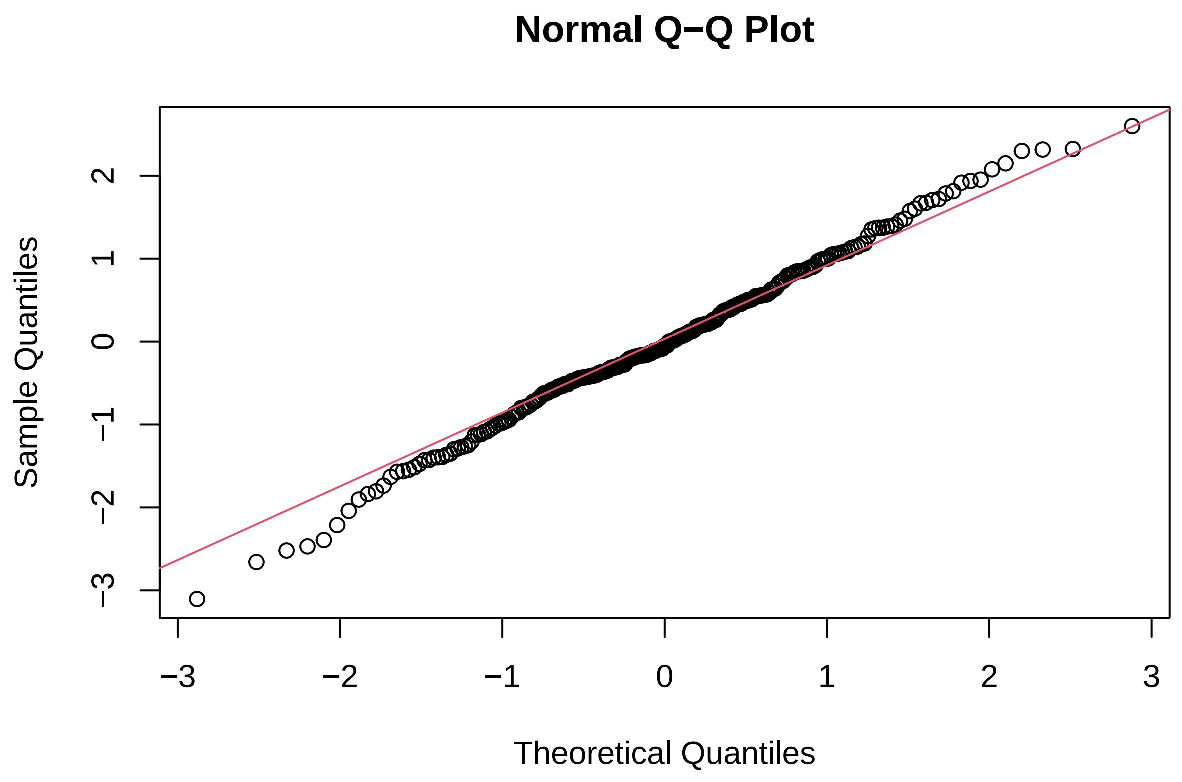
As expected, the MNAR case is more extreme in terms of the dissimilarities between two distributions.

(d) Since the process is still similar with (b), we only need to modify the name of some parameters or variables, we simplify the introduction here.

```
# construct data.frame
Y2_MNAR_na <- rep(NA,n)
ind_mnar_obs <- which(f_mnar >= 0)
for(i in ind_mnar_obs)
  Y2_MNAR_na[i] <- Y2[i]
# stochastic regression imputation
data2 <- data.frame("y1" = Y1, "y2" = Y2_MNAR_na)
fit2 <- lm(y2 ~ y1, data = data2)
# summary(fit2)
fit2
```

```
##
## Call:
## lm(formula = y2 ~ y1, data = data2)
##
## Coefficients:
## (Intercept)           y1
##       4.278        1.439
```

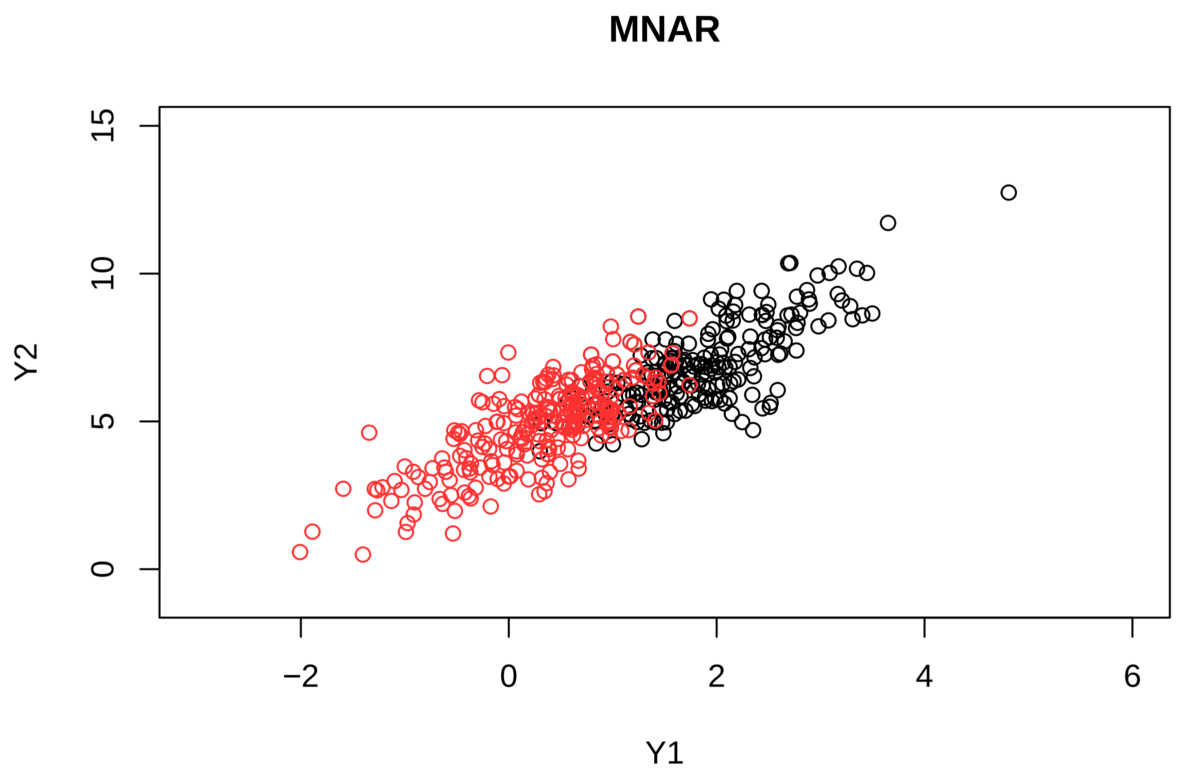
```
qqnorm(rstandard(fit2))
qqline(rstandard(fit2), col = 2)
```



```
set.seed(3)
predicted2_Y2 <- predict(fit2, newdata = data2) + rnorm(n, 0, sigma(fit2))
Y2_sri2 <- ifelse(is.na(Y2_MNAR_na), predicted2_Y2, Y2_MNAR_na)
```

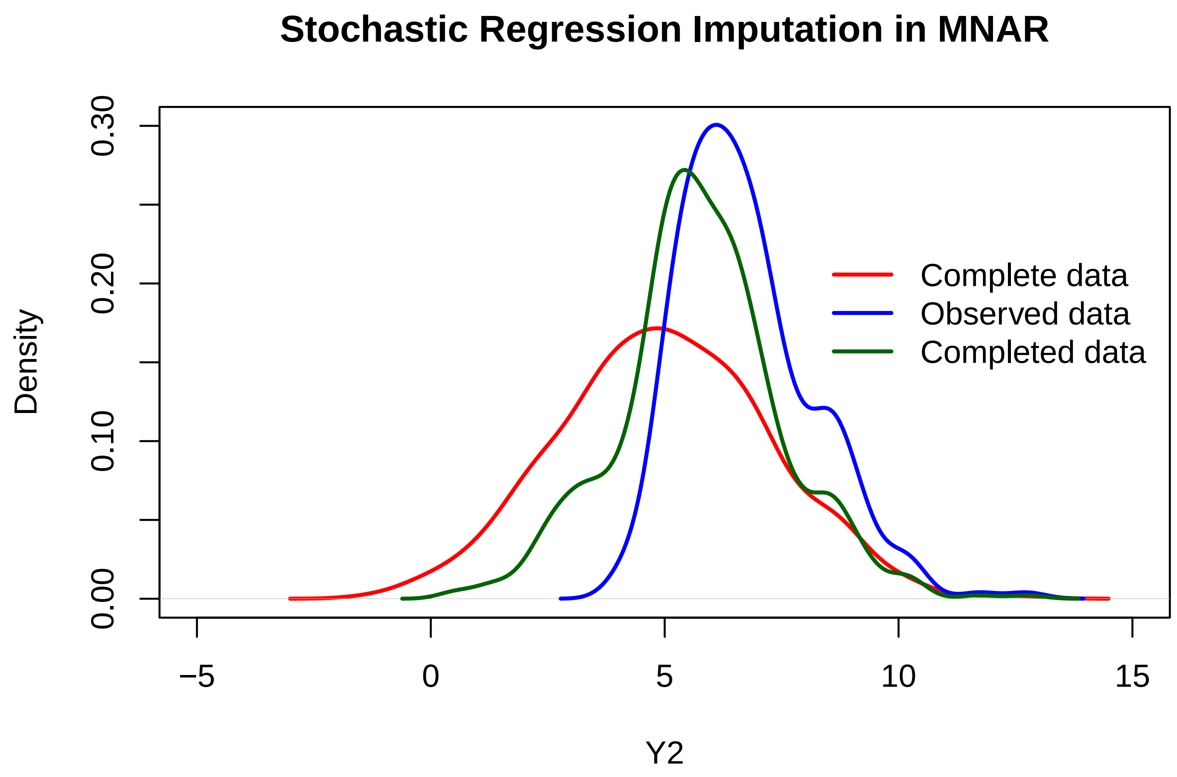
Note Although not required, we still show the scatter plot with the additional points in red.

```
# scatter plot
plot(x = data2$y1, y = data2$y2, xlab = "Y1", ylab = "Y2", xlim = c(-3,6),
     ylim = c(-1,15), main = "MNAR")
for(i in ind_mar_mis)
  points(x = Y1[i], y = Y2_sri2[i], col = "firebrick1")
```



Now, with comparison of complete data and biased observed data, we find that the distribution of imputed data by stochastic regression imputation fits the original shape well, as follows:

```
# distribution plot
plot(density(Y2),
     col = "red", ylim = c(0, 0.3), main = "Stochastic Regression Imputation in MNAR",
     xlab = "Y2", ylab = "Density", xlim = c(-5,15), lwd = 2)
lines(density(Y2_MNAR_obs), col = "blue", lwd = 2)
lines(density(Y2_sri2), col = "darkgreen", lwd = 2)
legend(8, 0.23, legend = c("Complete data", "Observed data", "Completed data"),
       col = c("red", "blue", "darkgreen"), lty = c(1, 1, 1), bty ="n", lwd = c(2,2,2))
```



Note that this time the imputation is not good enough. This is because that under the MNAR, the missing data pattern depends on the unobserved data and may depend also on the observed data (specifically, in this example, although a=0 and b=2 indicate that Y_1 is not considered in the missing mechanism, we should notice that, both Y_1 and Y_2 has the same component, Z_1, so that actually Y_1 can explain some of the missing mechanism, but in a limited and weak degree, resulting in a slight improvement from the observed Y_2 distribution shifting to the original complete one).

4.

(a) Under complete case analysis, we first compute the mean of recovery time.

```
# complete case analysis: mean value of recovery time
mean_recovtime_cc <- mean(databp$recovtime, na.rm = TRUE)
mean_recovtime_cc
```

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

Then compute the standard error of mean of recovery time following the formula:

$$\sigma_{reco\bar{v}time} = \frac{\sigma}{\sqrt{n}}.$$

```
# complete case analysis: standard error of mean of recovery time
se_recovtime_cc <- sd(databp$recovtime, na.rm = TRUE)/
  sqrt(length(databp$recovtime[-which(is.na(databp$recovtime)==TRUE)]))
se_recovtime_cc
```

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

When compute the pearson correlation between recovery time and dose, note that, in the dataset, only the recovery time contains missing values, so that in this example, `use = "complete"` and `use = "pairwise.complete.obs"` are equivalent with the same result.

```r
# complete case analysis: pearson correlation between recovery time and dose
cor_rd_cc <- cor(databp$recovtime, databp$logdose, use = "complete")
cor_rd_cc
```

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

The same computation process of pearson correlation, here between recovery time and blood pressure.

```r
# complete case analysis: pearson correlation between recovery time and blood pressure
cor_rb_cc <- cor(databp$recovtime, databp$bloodp, use = "complete")
cor_rb_cc
```

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

(b) Under mean imputation, we first compute the mean of recovery time (this step is completely the same as before).

```r
# mean imputation: mean value of recovery time
mean_recovtime_mi <- mean(databp$recovtime, na.rm = TRUE)
mean_recovtime_mi
```

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

The mean value keeps the same as the result in (a), because we do not change the mean, just repeat it for several times.

With the mean value of recovery time, now we can implement the mean imputation, by replacing all the NA into the obtained mean value. In the following computation, we will always use this completed revovery time rather than the one with missing value.

```r
# mean imputation implementation
recovtime_mi <- databp$recovtime
recovtime_mi[which(is.na(recovtime_mi)==TRUE)] <- mean_recovtime_mi
recovtime_mi[which(is.na(databp$recovtime)==TRUE)]
```

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

Then compute the standard error of mean of imputed recovery time following the formula as before, but now the n increases since missing values have been completed.

```r
# mean imputation: standard error of mean of recovery time
se_recovtime_mi <- sd(recovtime_mi)/sqrt(length(recovtime_mi))
se_recovtime_mi
```

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

The standard error of mean decreases, compared with the value in (a), as we repeated the mean value, the belief (confidence) increases and uncertainty decreases (as we "believe" the imputed value is true value now).

Since recovery time has been completed, the lengths between two variables are the same now, so that we do not need to apply the parameter `use` anymore, we just directly compute the correlation.

```r
# mean imputation: pearson correlation between recovery time and dose
cor_rd_mi <- cor(recovtime_mi, databp$logdose)
cor_rd_mi
```

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

The same process as last step above.

```
# mean imputation: pearson correlation between recovery time and blood pressure
cor_rb_mi <- cor(recovtime_mi, databp$bloodp)
cor_rb_mi
```

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

Note that the absolute values of both correlations between recovery time with other variables decrease, indicating a weaker relationship with others, because missing values (now completed) contribute with a value of zero to the numerator formula in covariance and correlation calculation.

(c) To implement the mean regression imputation, first we need to fit a regression model on the complete cases, using the recovery time as response variable and the other two variables as the covariate. Then with the model, we could predict the missing data with observations.

```
# mean regression imputation implementation
fit_mri <- lm(recovtime ~ logdose+bloodp, data = databp)
prediction_mri <- predict(fit_mri, newdata = databp)
recovtime_mri <- ifelse(is.na(databp$recovtime), prediction_mri, databp$recovtime)
recovtime_mri[which(is.na(databp$recovtime)==TRUE)]
```

```
## [1] 14.26254 21.51562 26.32896
```

Under mean regression imputation, now the mean of recovery time is also obtained from a complete data set, without missing value.

```
# mean regression imputation: mean value of recovery time
mean_recovtime_mri <- mean(recovtime_mri)
mean_recovtime_mri
```

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

Note that the mean value increases a little, compard with the result in (a).

Compute the standard error of mean of imputed recovery time following the formula as before, and now the n increases since missing values have been completed.

```
# mean regression imputation: standard error of mean of recovery time
se_recovtime_mri <- sd(recovtime_mri)/sqrt(length(recovtime_mri))
se_recovtime_mri
```

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

The standard error of mean decreases, compared with the value in (a), as this method will overestimate the correlations (higher confidence and stronger belief).

Since recovery time has been completed, the lengths between two variables are the same now, so that we do not need to apply the parameter `use` anymore, we just directly compute the correlation.

```
# mean regression imputation: pearson correlation between recovery time and dose
cor_rd_mri <- cor(recovtime_mri, databp$logdose)
cor_rd_mri
```

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

The same process as last step above.

```
# mean regression imputation: pearson correlation between recovery time and blood pressure
cor_rb_mri <- cor(recovtime_mri, databp$bloodp)
cor_rb_mri
```

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

Note that the absolute values of correlation between recovery time and dose increases dramatically while another correlation decreases a little. However, the overall correlation still increases.

(d) Here we can adopt the regression model directly from previous step, adding additional random residual term.

```r
# stochastic regression imputation implementation
set.seed(4)
prediction_sri <- predict(fit_mri, newdata = databp) +
  rnorm(length(databp$recovtime), 0, sigma(fit_mri))
recovtime_sri <- ifelse(is.na(databp$recovtime), prediction_sri, databp$recovtime)
recovtime_sri[which(is.na(databp$recovtime)==TRUE)]
```

```
## [1] 21.56488 43.28692 28.35272
```

We need to be careful here, since the regression predictions with random residuals may be negative (they are not restricted), but intuitively, the recovery time could not be a negative value. Fortunately, here the imputed values are all positive, so we do not need to do further operation.

Under stochastic regression imputation, now the mean of recovery time is also obtained from a complete data set, without missing value.

```r
# stochastic regression imputation: mean value of recovery time
mean_recovtime_sri <- mean(recovtime_sri)
mean_recovtime_sri
```

```
## [1] 20.68818
```

Note that the mean value increases a little, compard with the results in both (a) and (c).

Compute the standard error of mean of imputed recovery time following the formula as before.

```r
# stochastic regression imputation: standard error of mean of recovery time
se_recovtime_sri <- sd(recovtime_sri)/sqrt(length(recovtime_sri))
se_recovtime_sri
```

```
## [1] 2.498219
```

The standard error of mean decreases, compared with the value in (a), while increases compared with result in (c).

Since recovery time has been completed, the lengths between two variables are the same now, so that we do not need to apply the parameter `use` anymore, we just directly compute the correlation.

```r
# stochastic regression imputation: pearson correlation between recovery time and dose
cor_rd_sri <- cor(recovtime_sri, databp$logdose)
cor_rd_sri
```

```
## [1] 0.3008403
```

The same process as last step above.

```r
# stochastic regression imputation: pearson correlation between recovery time and blood pressure
cor_rb_sri <- cor(recovtime_sri, databp$bloodp)
cor_rb_sri
```

```
## [1] 0.03706711
```

Note that both the absolute values of correlations between recovery time and others increases compared with results of (a) and (c).

(e) Here we can adopt the regression model directly from mean regression imputation step by texttt{prediction_mri}, and follow the criterion called predictive mean matching referrer to Little (1988). The formula is as below:

$$\hat{recovtime_j} = recovtime_k,$$

where $(\hat{\mu}_j - \hat{\mu}_k)^2 \leq (\hat{\mu}_j - \hat{\mu}_l)^2$ for all respondents l, $\hat{\mu}_j$ is the predicted mean of revovery time for individual j, and $recovtime_k$ is the observed value of recovery time for respondent k.

```
# predictive mean matching implementation
prediction_pmm <- prediction_mri
y_true <- databp$recovtime[-which(is.na(databp$recovtime)==TRUE)]
for(j in which(is.na(databp$recovtime)==TRUE))
{
  k <- which.min((prediction_pmm[j]-y_true)^2)
  prediction_pmm[j] <- y_true[k]
}
recovtime_pmm <- ifelse(is.na(databp$recovtime), prediction_pmm, databp$recovtime)
recovtime_pmm[which(is.na(databp$recovtime)==TRUE)]
```

```
## [1] 14 22 28
```

Note that now all the replacement of missing value are "selected" from the observed (existed) data. This is a beneficial property, because all the imputed data has already existed, so that they will obey the potential data pattern restriction and limitation. For example, the regression-based methods may lead a negative value of recovery time (although not happen yet), but "time" could not be negative in this problem. However, if we use the predictive mean matching method, even a predicted missing value is negative, then we will match a small positive observed value as replacement.

With a completed data set, now we calculate the mean directly.

```
# predictive mean matching: mean value of recovery time
mean_recovtime_pmm <- mean(recovtime_pmm)
mean_recovtime_pmm
```

```
## [1] 19.52
```

Compared with the mean in (a) and (c), here the mean is larger; but it is smaller than the mean in (d).

Compute the standard error of mean of imputed recovery time following the formula as before.

```
# predictive mean matching: standard error of mean of recovery time
se_recovtime_pmm <- sd(recovtime_pmm)/sqrt(length(recovtime_pmm))
se_recovtime_pmm
```

```
## [1] 2.323876
```

The standard error of mean decreases, compared with the values in (a) and (d), while increases compared with result in (c).

Since recovery time has been completed, the lengths between two variables are the same now, so that we do not need to apply the parameter `use` anymore, we just directly compute the correlation.

```
# predictive mean matching: pearson correlation between recovery time and dose
cor_rd_pmm <- cor(recovtime_pmm, databp$logdose)
cor_rd_pmm
```

```
## [1] 0.2908267
```

The same process as last step above.

```r
# predictive mean matching: pearson correlation between recovery time and blood pressure
cor_rb_pmm <- cor(recovtime_pmm, databp$bloodp)
cor_rb_pmm
```

```
## [1] -0.009406132
```

Here we choose the result in (c) to compare. We get a larger correlation between recovery time and dose here, while a weaker correlation between recovery time and blood pressure.

(f)

Advantage of predictive mean matching over stochastic regression imputation: As all the imputed data of predictive mean matching are "selected" from the observed (existed) data set, it will keep the same restriction or pattern as observation or prior knowledge, such as non-negativity, binary distribution, an so on. As an example, if we obtain a negative value from the first step regression, but the data itself is non-negative, then the smallest observed (positive) value would be matched. However, in stochastic regression, predicted data may violate the data restrictions.

Potential problem of predictive mean matching: if the data set is sparse (specifically, large range while few data, each one is far away from others), then we may cannot find a close enough observation, leading to a weak match. It may produce substantially biased estimates of correlations and regression coefficients. Again, like any other single imputation procedure, it may underestimate standard errors (although some corresponding corrections have been proposed).