Assignment 1

Ted Ladas -s2124289

10/20/2020

Github repo: https://github.com/TedOiler/ida_assignment_1

Exercise 1

-(a) –> (ii) We are told that the missing mechanism is MCAR. That means that the missingness doesn't depend on any other variable.

-(b) –> (ii) This is because if the mechanism is MAR, then the probability of ALQ being missing depends on some variable that has fully observed values. I this case it's the gender as per the exercise definition

-(c) –> (iii) From the information given, the only thing that we can infer is that the probability of ALQ not missing given that gender is Male is 0.9. We cannot conclude about the probability on all the other strata of gender.

Exercise 2

Largest possible subsample: 90 rows. That will happen when each of the 10 variables, are missing exactly the same rows (let suppose, without loss of generality rows 1:10)

Smallest possible subsample: 0 rows. In the extreme case where there is no overlap on any rows between the 10 variables, the dataset produced where each row has information on all variables will be of 0 length. Suppose X1 is missing 1:10, X2 is missing 11:21, . . . , X10: 90:100

Exercise 3

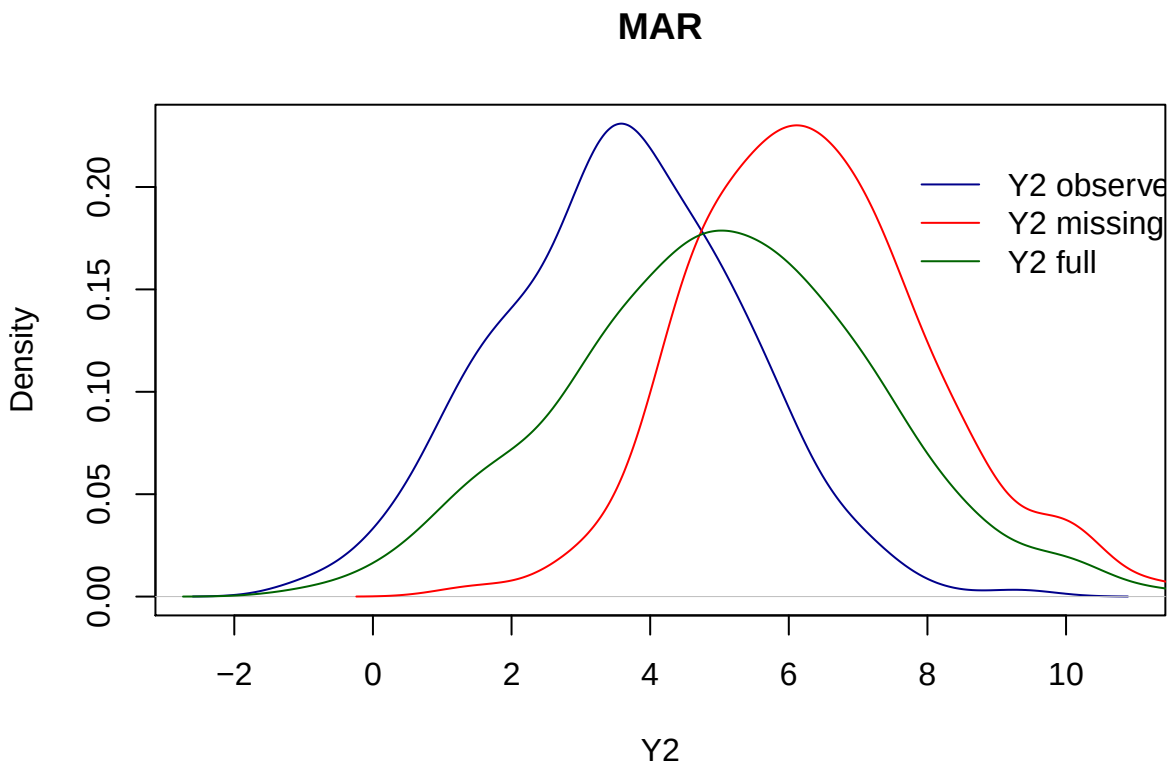
Setup of full dataframe

```
set.seed(1903)
rows = 500
z1 = rnorm(rows)
z2 = rnorm(rows)
z3 = rnorm(rows)

y1 = 1 + z1
y2 = 5 + 2 * z1 + z2
df = data.frame(Y1 = y1, Y2 = y2)
```

-3.a

Comments: The mechanism of missingness is MAR because the missingness of the variable Y2 depends on the Y1 variable with a=2 and b=0. On the diagram we observe the shift in position of the observed data versus the full dataset.

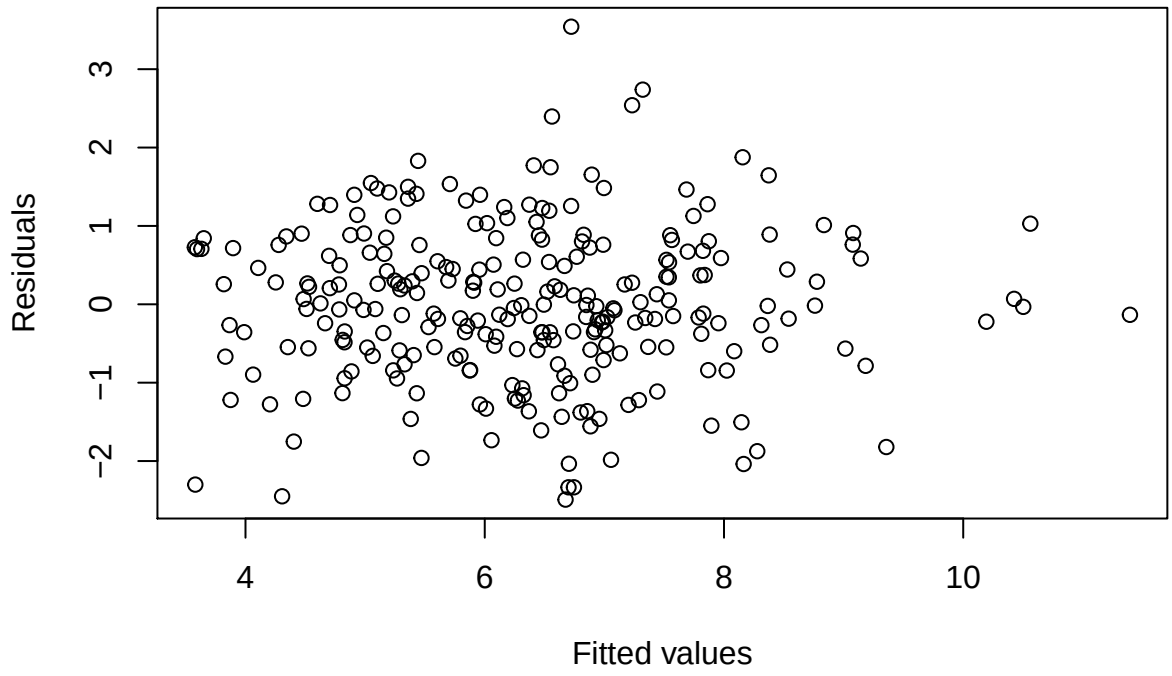
```
df_1 = df
a = 2
b = 0
y2_rule = a * (y1 - 1) + b * (y2 - 5) + z3 < 0

y2_obs = y2[y2_rule]
y2_miss = y2[!y2_rule]

for (i in 1:rows) {
    if (y2_rule[i]) {
        df_1$Y2[i] = NA
    }
}

plot(density(y2_obs), col = "darkblue", xlab = "Y2", main = "MAR")
lines(density(y2_miss), col = "red")
lines(density(df$Y2), col = "darkgreen")
legend(7.5, 0.22, legend = c("Y2 observed", "Y2 missing", "Y2 full"), col = c("darkblue",
    "red", "darkgreen"), lty = c(1, 1, 1), bty = "n")
```



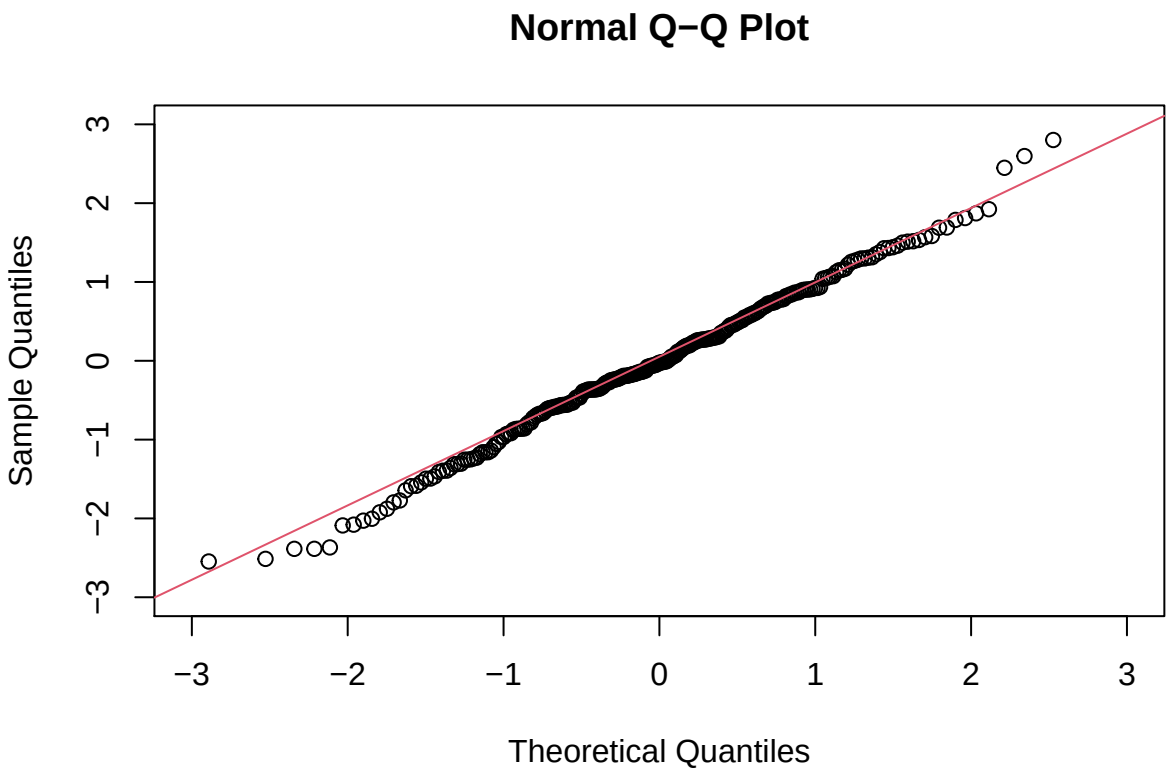
-3.b

Comments about checks of assumption: First I am checking for any correlation between the residuals of the regression and the Y2 variable, and whether the residuals actually are distributed identically and independently distributed, with constant variance.I see that this is the case because on the first diagram there are no obvious (at least to the eye) pattern and on the second diagram I see that the they all fall onto the line. I suspected that this will be the case because this is a generated dataset from Normal distributions, meaning that the assumptions of OLS will hold perfectly.

```
fit = lm(df_1$Y2 ~ Y1, data = df_1)
# checking for correlation between Y2 and residuals
plot(fit$fitted.values, residuals(fit), xlab = "Fitted values", ylab = "Residuals")
```




```
# checking for irregularities in the distribution of residuals
qqnorm(rstandard(fit), xlim = c(-3, 3), ylim = c(-3, 3))
qqline(rstandard(fit), col = 2)
```

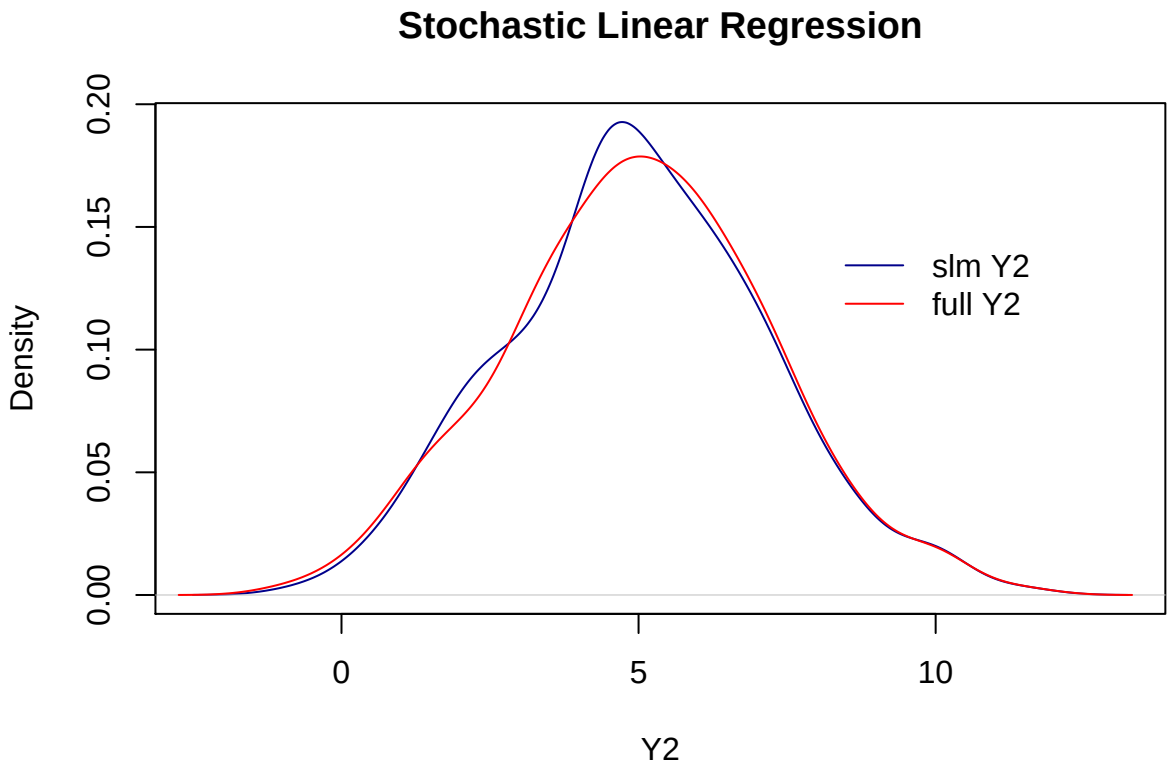



```
summary(fit)
```

```
##
## Call:
## lm(formula = df_1$Y2 ~ Y1, data = df_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4918 -0.5720 -0.0243  0.6723  3.5421
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.9802     0.1571   18.97   <2e-16 ***
## Y1            1.9873     0.0860   23.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9806 on 259 degrees of freedom
##   (239 observations deleted due to missingness)
## Multiple R-squared:  0.6734, Adjusted R-squared:  0.6721
## F-statistic:   534 on 1 and 259 DF,  p-value: < 2.2e-16
```

Comments about actual results: As expected, the Stochastic Linear Regression imputation, does a very good job on imputing the missing values, since we saw above that all the assumptions hold.

```r
df_2 = df_1
predicted_slm = predict(fit, newdata = df_1) + rnorm(nrow(df_1), 0, sigma(fit))
df_2$Y2 = ifelse(is.na(df_2$Y2), predicted_slm, df_2$Y2)

plot(density(df_2$Y2), col = "darkblue", xlab = "Y2", main = "Stochastic Linear Regression")
lines(density(df$Y2), col = "red")
legend(8, 0.15, legend = c("slm Y2", "full Y2"), col = c("darkblue", "red"), lty = c(1,
    1), bty = "n")
```

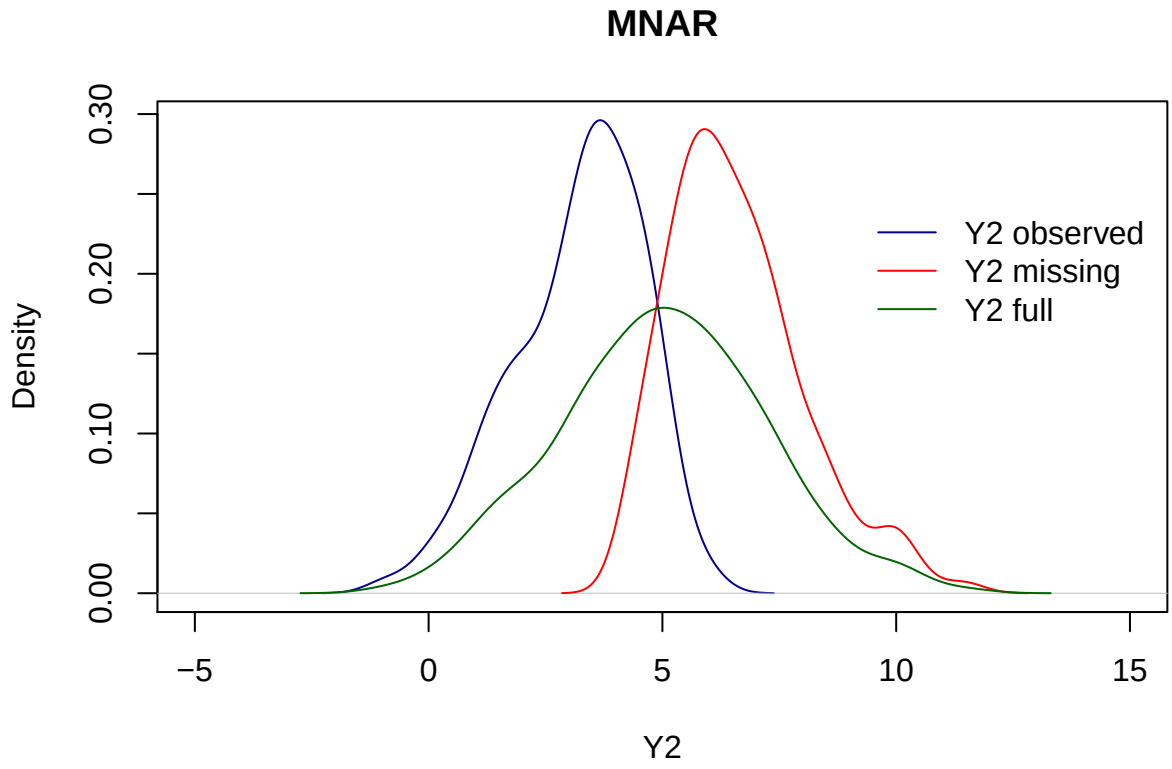


-3.c

Comments: The mechanism of missingness is MNAR because the missingness of the variable Y2 depends on the Y2 itselft with a=0 and b=2. On the diagram we observe the shift in position of the observed data versus the full dataset, as well as the change in standard error caused by the specific mechanism.

```r
df_3 = df
a = 0
b = 2
y2_rule = a * (y1 - 1) + b * (y2 - 5) + z3 < 0

y2_obs = y2[y2_rule]
y2_miss = y2[!y2_rule]

for (i in 1:rows) {
```

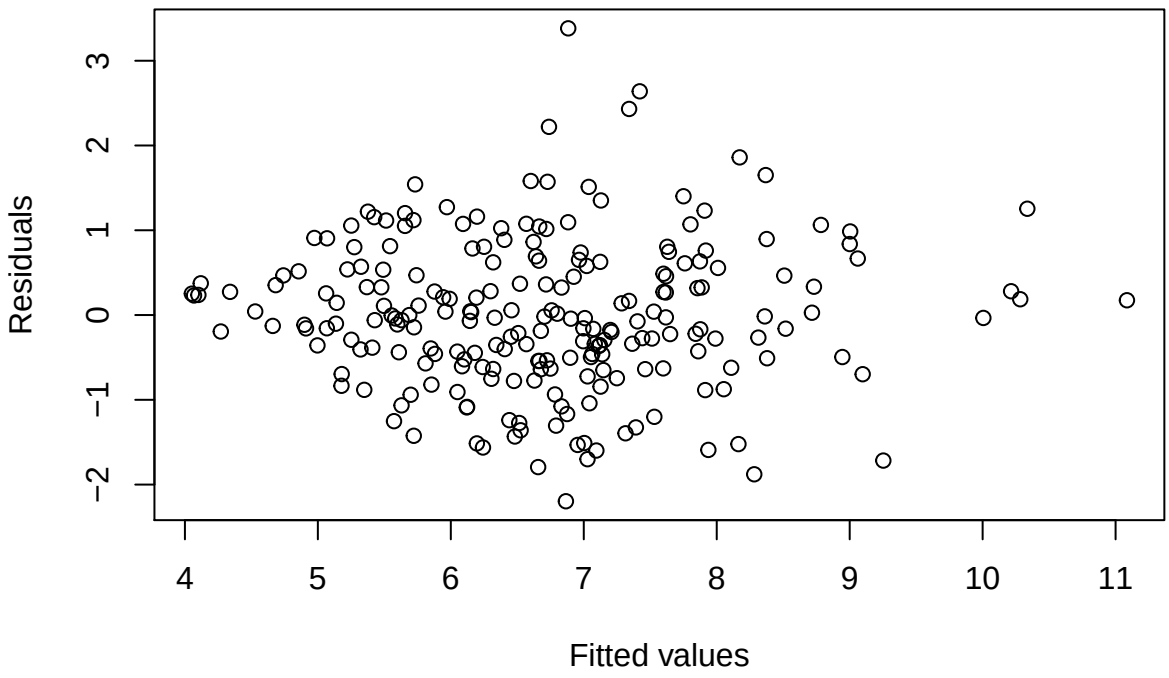
```
    if (y2_rule[i]) {
        df_1$Y2[i] = NA
    }
}

plot(density(y2_obs), col = "darkblue", xlab = "Y2", main = "MNAR", xlim = c(-5,
    15))
lines(density(y2_miss), col = "red")
lines(density(df$Y2), col = "darkgreen")
legend(9, 0.25, legend = c("Y2 observed", "Y2 missing", "Y2 full"), col = c("darkblue",
    "red", "darkgreen"), lty = c(1, 1, 1), bty = "n")
```



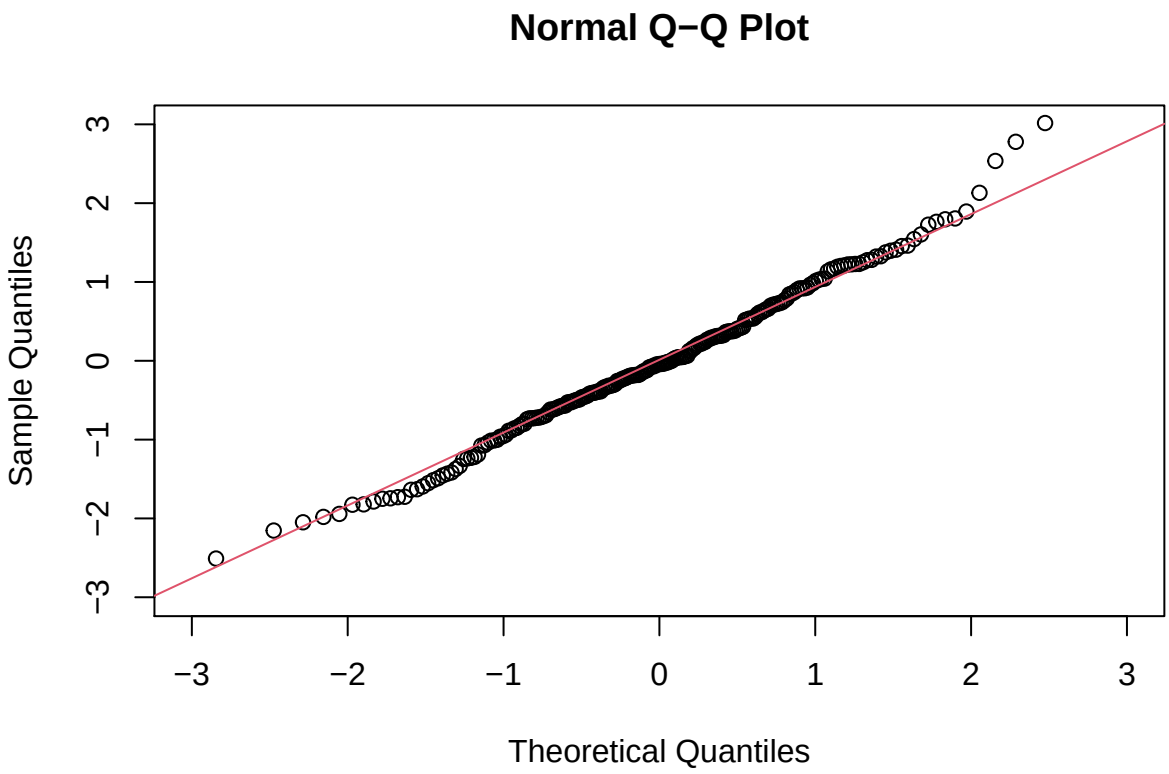
-3.d

Comments about checks of assumption: As in 3.b the assumptions still hold for the most part, since those check are dependent on the original distribution of the data which is unchanged.

```
fit = lm(df_1$Y2 ~ df_3$Y1, data = df_1)
# checking for correlation between Y2 and residuals
plot(fit$fitted.values, residuals(fit), xlab = "Fitted values", ylab = "Residuals")
```




```r
# checking for irregularities in the distribution of residuals
qqnorm(rstandard(fit), xlim = c(-3, 3), ylim = c(-3, 3))
qqline(rstandard(fit), col = 2)
```

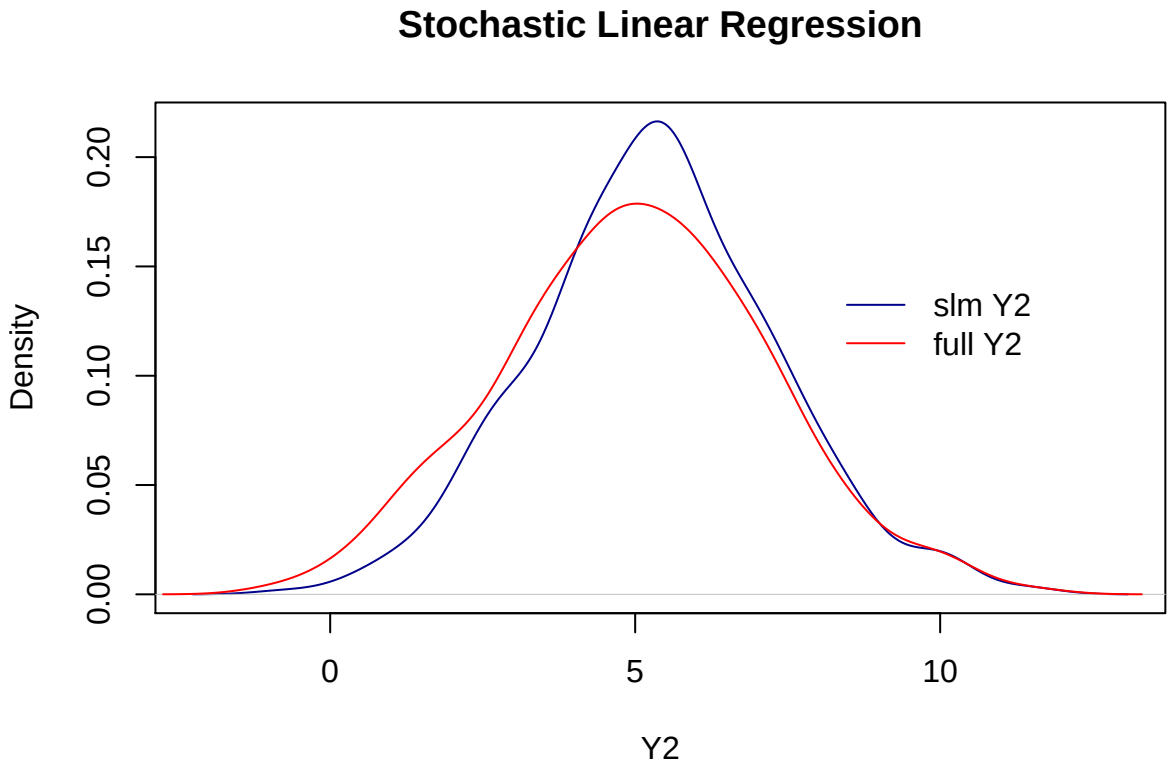



```
summary(fit)
```

```
##
## Call:
## lm(formula = df_1$Y2 ~ df_3$Y1, data = df_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1967 -0.5345 -0.0361  0.5555  3.3820
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.51555    0.16340   21.52   <2e-16 ***
## df_3$Y1      1.78801    0.08574   20.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8774 on 223 degrees of freedom
##   (275 observations deleted due to missingness)
## Multiple R-squared:  0.661,  Adjusted R-squared:  0.6595
## F-statistic: 434.9 on 1 and 223 DF,  p-value: < 2.2e-16
```

Comments about actual results: As with the case of the MAR, the stochastic linear regression does a good job at imputing the values since the results are unbiased. However, we can observe from the diagram the fall in efficiency, since our standard deviation is skewed.

```
df_4 = df_1
predicted_slm = predict(fit, newdata = df_1) + rnorm(nrow(df_1), 0, sigma(fit))
df_4$Y2 = ifelse(is.na(df_4$Y2), predicted_slm, df_4$Y2)

plot(density(df_4$Y2), col = "darkblue", xlab = "Y2", main = "Stochastic Linear Regression")
lines(density(df$Y2), col = "red")
legend(8, 0.15, legend = c("slm Y2", "full Y2"), col = c("darkblue", "red"), lty = c(1,
    1), bty = "n")
```



Exercise 4

-4.a

```
load(file = "databp.RData")
N = length(databp$recovtime)
mean_fc = mean(databp$recovtime, na.rm = TRUE)
se_fc = sd(databp$recovtime, na.rm = TRUE)/sqrt(N)
corr_recovtime_logdose_fc = cor(databp$recovtime, databp$logdose, use = "complete")
corr_recovtime_boodp_fc = cor(databp$recovtime, databp$bloodp, use = "complete")

print("Complete Case")
print(paste0("mean: ", mean_fc))
```

```
print(paste0("standard error: ", se_fc))
print(paste0("recovtime & dose: ", corr_recovtime_logdose_fc))
print(paste0("recovtime & bloodp: ", corr_recovtime_boodp_fc))
```

```
## [1] "Complete Case"
## [1] "mean: 19.2727272727273"
## [1] "standard error: 2.44184303398015"
## [1] "recovtime & dose: 0.23912557669433"
## [1] "recovtime & bloodp: -0.0195286212600658"
```

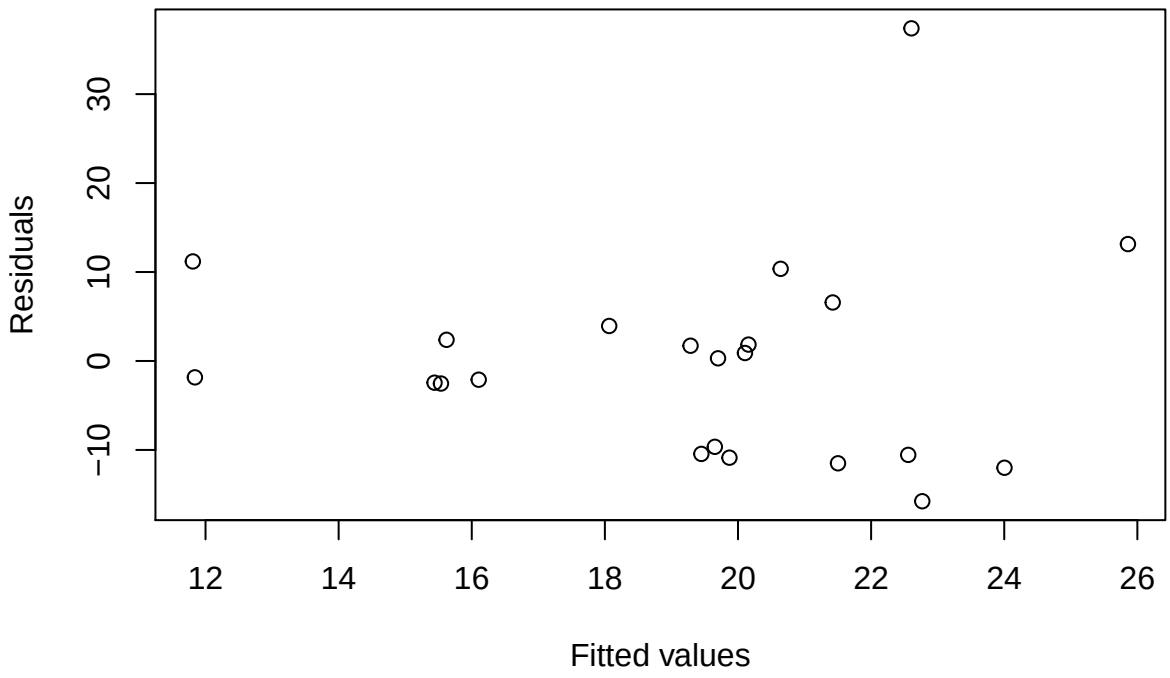
-4.b

```
recovtime_mi = ifelse(is.na(databp$recovtime), mean_fc, databp$recovtime)

mean_mi = mean(recovtime_mi)
se_mi = sd(recovtime_mi)/sqrt(N)
corr_recovtime_logdose_mi = cor(recovtime_mi, databp$logdose)
corr_recovtime_boodp_mi = cor(recovtime_mi, databp$bloodp)

print("Mean imputation")
print(paste0("mean: ", mean_mi))
print(paste0("standard error: ", se_mi))
print(paste0("recovtime & dose: ", corr_recovtime_logdose_mi))
print(paste0("recovtime & bloodp: ", corr_recovtime_boodp_mi))
```

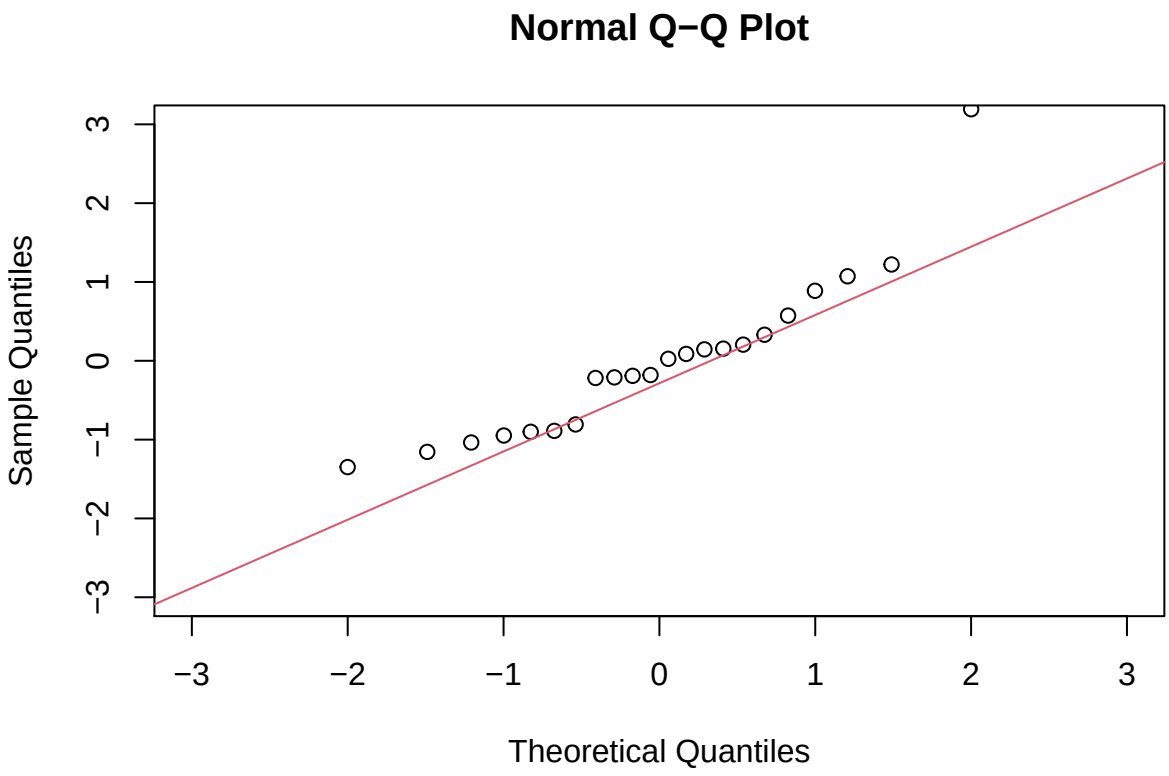
```
## [1] "Mean imputation"
## [1] "mean: 19.2727272727273"
## [1] "standard error: 2.28413500635858"
## [1] "recovtime & dose: 0.215061174129736"
## [1] "recovtime & bloodp: -0.0193412641944137"
```

-4.c

```
fit = lm(databp$recovtime ~ databp$logdose + databp$bloodp, data = databp)

# checking for correlation between Y2 and residuals
plot(fit$fitted.values, residuals(fit), xlab = "Fitted values", ylab = "Residuals")
```




```
# checking for irregularities in the distribution of residuals
qqnorm(rstandard(fit), xlim = c(-3, 3), ylim = c(-3, 3))
qqline(rstandard(fit), col = 2)
```




```
# summary(fit) Rsqr seems very bad, need to explore more
predicted_ri = predict(fit, newdata = databp)
recovtime_ri = ifelse(is.na(databp$recovtime), predicted_ri, databp$recovtime)

mean_ri = mean(recovtime_ri)
se_ri = sd(recovtime_ri)/sqrt(N)
corr_recovtime_logdose_ri = cor(recovtime_ri, databp$logdose)
corr_recovtime_boodp_ri = cor(recovtime_ri, databp$bloodp)

print("Regression Imputation")
print(paste0("mean: ", mean_ri))
print(paste0("standard error: ", se_ri))
print(paste0("recovtime & dose: ", corr_recovtime_logdose_ri))
print(paste0("recovtime & bloodp: ", corr_recovtime_boodp_ri))
```

```
## [1] "Regression Imputation"
## [1] "mean: 19.4442847814827"
## [1] "standard error: 2.31284487743953"
## [1] "recovtime & dose: 0.280183502439952"
## [1] "recovtime & bloodp: -0.0111364040939202"
```

-4.d

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

mean_sri = mean(recovtime_sri)
se_sri = sd(recovtime_sri)/sqrt(N)
corr_recovtime_logdose_sri = cor(recovtime_sri, databp$logdose)
corr_recovtime_boodp_sri = cor(recovtime_sri, databp$bloodp)

print("Stochastic Regression Imputation")
print(paste0("mean: ", mean_sri))
print(paste0("standard error: ", se_sri))
print(paste0("recovtime & dose: ", corr_recovtime_logdose_sri))
print(paste0("recovtime & bloodp: ", corr_recovtime_boodp_sri))

# very skewed results, need to investigate
```

```
## [1] "Stochastic Regression Imputation"
## [1] "mean: 18.9203753875029"
## [1] "standard error: 2.45562932564445"
## [1] "recovtime & dose: 0.302186205262322"
## [1] "recovtime & bloodp: 0.0082648700658505"
```

-4.e

Comments: As we can see, the correlation between logdose, and bloodp is above 50%! This means that on our regression model we have multicollinearity issues. This violates the assumptions of OLS. The OLS still produces some results since we do not have perfect collinearity but the results are not that statistically significant as in for example Exercise 3. To overcome this problem we could try to do our regression with either of the variables, but this might oversimplify our model. At this point something worth mentioning is the fact that the correlation between recovery time and bloop pressure changes sign, which is alarming. This is due to the above mentioned problems of the regression assumptions.

```r
predicted_hd = predict(fit, newdata = databp)
ind_hd = which(is.na(databp$recovtime) == FALSE)

mean_sqr_diff = data.frame(data.frame(matrix(0, ncol = 3, nrow = 22)))
sqr_diff = data.frame(data.frame(matrix(0, ncol = 1, nrow = 3)))
recovtime_hd = c(databp$recovtime[is.na(databp$recovtime) == FALSE])

for (i in 1:length(predicted_hd[-ind_hd])) {
    mean_sqr_diff[, i] = (predicted_hd[ind_hd] - predicted_hd[-ind_hd][i])^2
    sqr_diff[i, ] = which.min(mean_sqr_diff[, i])
}

for (i in 1:length(predicted_hd[-ind_hd])) {
    recovtime_hd = append(recovtime_hd, databp$recovtime[sqr_diff[i, ]])
}

mean_hd = mean(recovtime_hd, na.rm = TRUE)
se_hd = sd(recovtime_hd, na.rm = TRUE)/sqrt(N)
corr_recovtime_logdose_hd = cor(recovtime_hd, databp$logdose)
corr_recovtime_boodp_hd = cor(recovtime_hd, databp$bloodp)
```

```r
corr_logdose_boodp_hd = cor(databp$logdose, databp$bloodp)

print("Predictive Mean Matching - Hot Deck Imputation")
print(paste0("logdose & bloodp correlation: ", corr_logdose_boodp_hd))
print(paste0("mean: ", mean_hd))
print(paste0("standard error: ", se_hd))
print(paste0("recovtime & dose: ", corr_recovtime_logdose_hd))
print(paste0("recovtime & bloodp: ", corr_recovtime_boodp_hd))
```

```
## [1] "Predictive Mean Matching - Hot Deck Imputation"
## [1] "logdose & bloodp correlation: 0.514315466074099"
## [1] "mean: 18.28"
## [1] "standard error: 2.35125498404575"
## [1] "recovtime & dose: -0.1652931751014"
## [1] "recovtime & bloodp: -0.202530843015088"
```

-4.f

Comments: Hot deck imputation has the advantage of using a simple linear regression model without the stochasticity and the drawing conclusions by imputing real (observed) values on the missing. Since we have seen on question 4.e that the normal qq plot suggests that our residuals are not iid, it would be wrong to use stochastic linear regression method of imputation. However one shortcoming that we see, is that the correlation of our dependent with our independent variables are 'overestimated' compared to the complete case analysis. Similarly we can detect a small bias on the mean values, when comparing with the mean of the complete case analysis.