Incomplete Data Analysis, 2020/2021
Single imputation – toy example

Vanda Inacio

I will first create the toy dataset containing the data on pain and depression scores.

```
data <- data.frame(PS = c(4, 6, 7, 7, 8, 9, 9, 10, 10, 11, 12, 14, 14, 14, 15, 16,
    16, 17, 18, 23), DEP = c(NA, NA, 14, 11, NA, NA, 11, NA, 16, 9, 9, 14, 16, 21,
    14, 14, 18, 19, 21, 18))

data
```

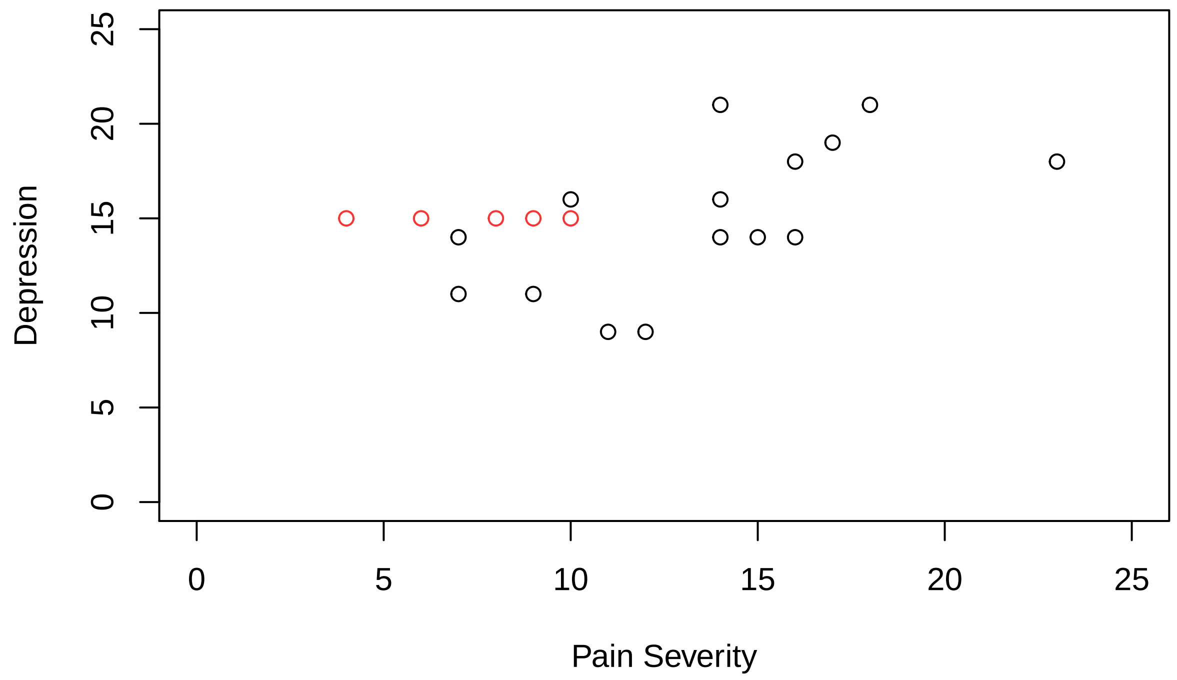
```
##    PS DEP
## 1   4  NA
## 2   6  NA
## 3   7  14
## 4   7  11
## 5   8  NA
## 6   9  NA
## 7   9  11
## 8  10  NA
## 9  10  16
## 10 11   9
## 11 12   9
## 12 14  14
## 13 14  16
## 14 14  21
## 15 15  14
## 16 16  14
## 17 16  18
## 18 17  19
## 19 18  21
## 20 23  18
```

To perform mean imputation, we just need to compute the mean of the observed depression scores.

```
mean(data$DEP, na.rm = TRUE)
```

```
## [1] 15
```

```
plot(data$PS, data$DEP, xlim = c(0, 25), ylim = c(0, 25), xlab = "Pain Severity",
    ylab = "Depression")
points(4, 15, col = "firebrick1")
points(6, 15, col = "firebrick1")
points(8, 15, col = "firebrick1")
points(9, 15, col = "firebrick1")
points(10, 15, col = "firebrick1")
```



```
# creating a vector with DEP values after mean imputation
DEP_mi <- ifelse(is.na(data$DEP), 15, data$DEP)

# standard deviation of the completed variable
sd(DEP_mi)
```

```
## [1] 3.371709
```

```
# standard deviation of the observed data on DEP variable
sd(data$DEP, na.rm = TRUE)
```

```
## [1] 3.927922
```

```
# same but for the correlation between pain and depression scores
cor(data$DEP, data$PS, use = "complete")
```

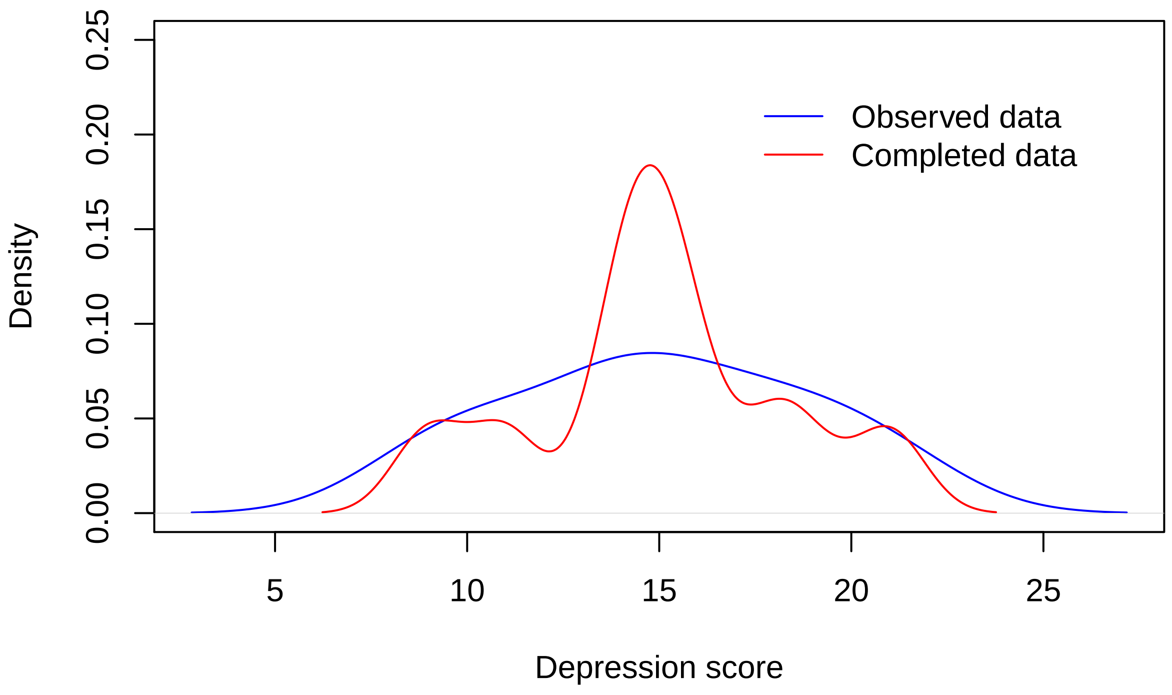
```
## [1] 0.6158968
```

```
cor(DEP_mi, data$PS)
```

```
## [1] 0.4834689
```

Let us also look at the (kernel) density estimates of the completed and observed variables.

```
plot(density(data$DEP[-which(is.na(data$DEP) == TRUE)]), col = "blue", ylim = c(0,
    0.25), main = "", xlab = "Depression score", ylab = "Density")
lines(density(DEP_mi), col = "red")
legend(17, 0.23, legend = c("Observed data", "Completed data"), col = c("blue", "red"),
    lty = c(1, 1), bty = "n")
```



We will now move to the conditional mean/regression imputation case. To do so, we first need to fit a regression model on the complete cases, using the depression score (variable with missing values) as the response variable and the pain severity as the covariate. We will fit a simple linear regression model of the form

$$\mathrm{DEP}_i = \beta_0 + \beta_1 \mathrm{PS}_i + \varepsilon_i, \qquad \varepsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2), \qquad i = 1, \ldots, 15,$$

which can be implemented using the `lm` command (and we already know from past lectures that this function automatically excludes all cases with missing values from the analysis).

```
fit <- lm(DEP ~ PS, data = data)
summary(fit)
```

```
##
## Call:
## lm(formula = DEP ~ PS, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1453 -2.0470 -0.3584  2.3547  5.7399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.4568     2.8016   2.662   0.0196 *
## PS            0.5574     0.1977   2.819   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.211 on 13 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.3793, Adjusted R-squared:  0.3316
## F-statistic: 7.945 on 1 and 13 DF,  p-value: 0.0145
```

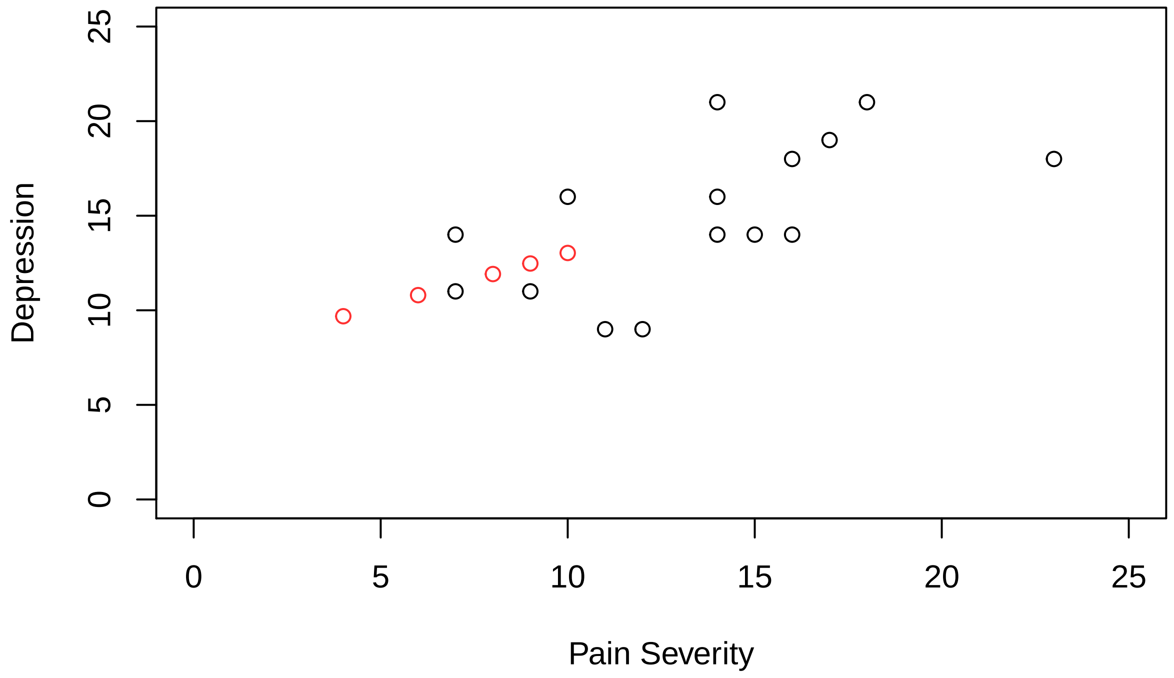
```
fit$coefficients
```

```
## (Intercept)          PS
##   7.4567745   0.5573812
```

We get the following estimates: $\widehat{\beta}_0 = 7.4568$ and $\widehat{\beta}_1 = 0.5574$. We can compute the predictions manually (as it was demonstrated in class), or we can do that automatically with the aid of the function `predict`.

```r
# predicting the DEP score for those with missing values
predicted_ri <- predict(fit, newdata = data)

plot(data$PS, data$DEP, xlim = c(0, 25), ylim = c(0, 25), xlab = "Pain Severity",
    ylab = "Depression")
points(4, predicted_ri[1], col = "firebrick1")
points(6, predicted_ri[2], col = "firebrick1")
points(8, predicted_ri[5], col = "firebrick1")
points(9, predicted_ri[6], col = "firebrick1")
points(10, predicted_ri[8], col = "firebrick1")
```



```r
# creating the completed DEP variable
DEP_ri <- ifelse(is.na(data$DEP), predicted_ri, data$DEP)

# std deviation for the observed and completed data
sd(data$DEP, na.rm = TRUE)
```

```
## [1] 3.927922
```

```r
sd(DEP_ri)
```

```
## [1] 3.748918
```

```r
# correlation for the observed and completed data
cor(data$DEP, data$PS, use = "complete")
```

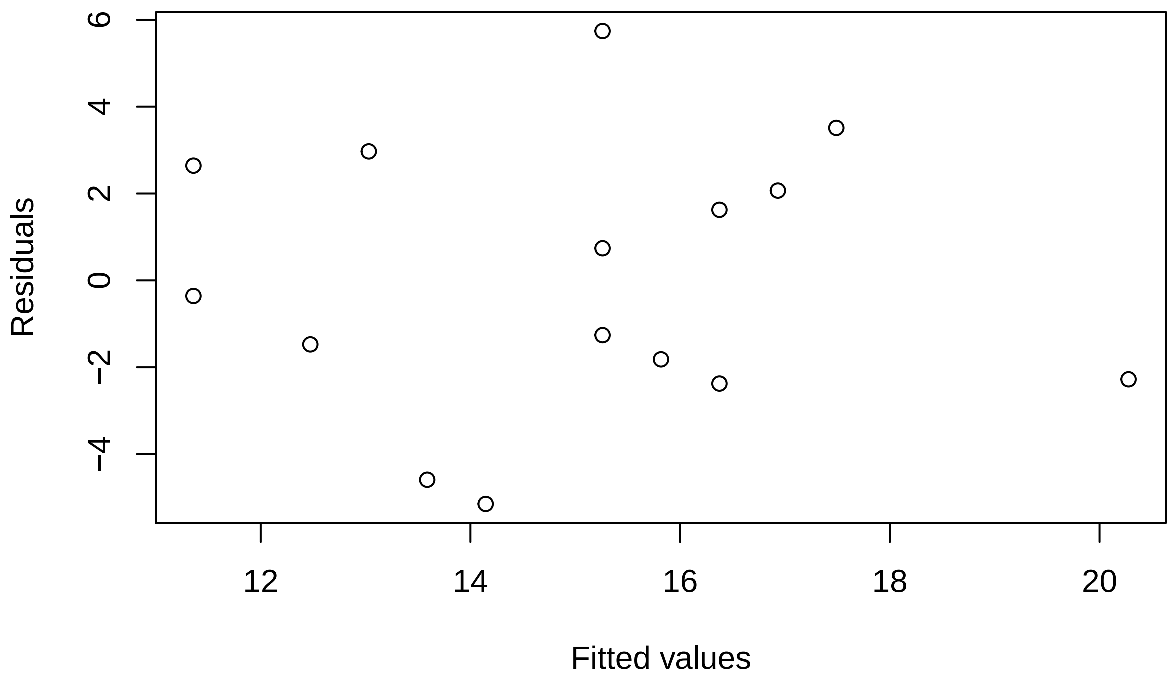
```
## [1] 0.6158968
```

```r
cor(DEP_ri, data$PS)
```

```
## [1] 0.7056534
```

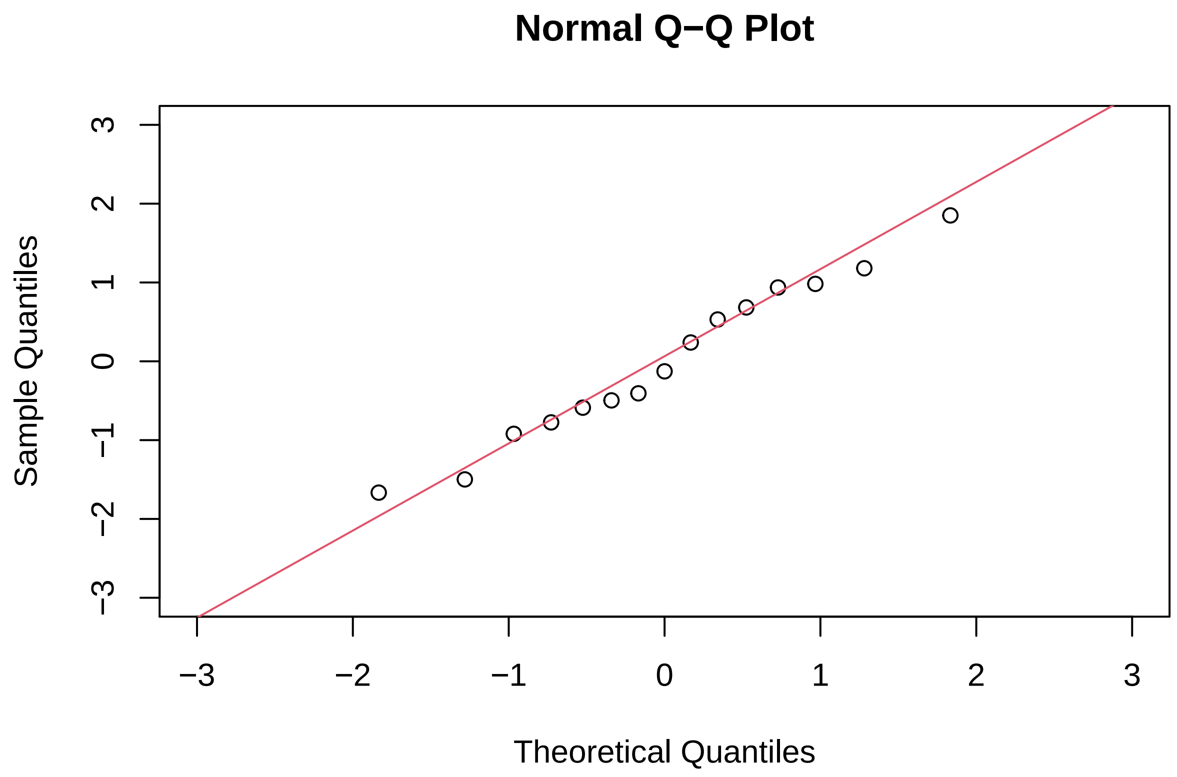
To check the assumption of linearity (between pain and depression scores) and of constant variance we can use a plot of the fitted values against the residuals.

```r
plot(fit$fitted.values, residuals(fit), xlab = "Fitted values", ylab = "Residuals")
```



There is no obvious pattern, so there is no reason to suspect of a nonlinear relationship or no constant variance. Although it is not of fundamental importance here in regression imputation, we can also check the assumption of normality of error terms using a QQ-plot.

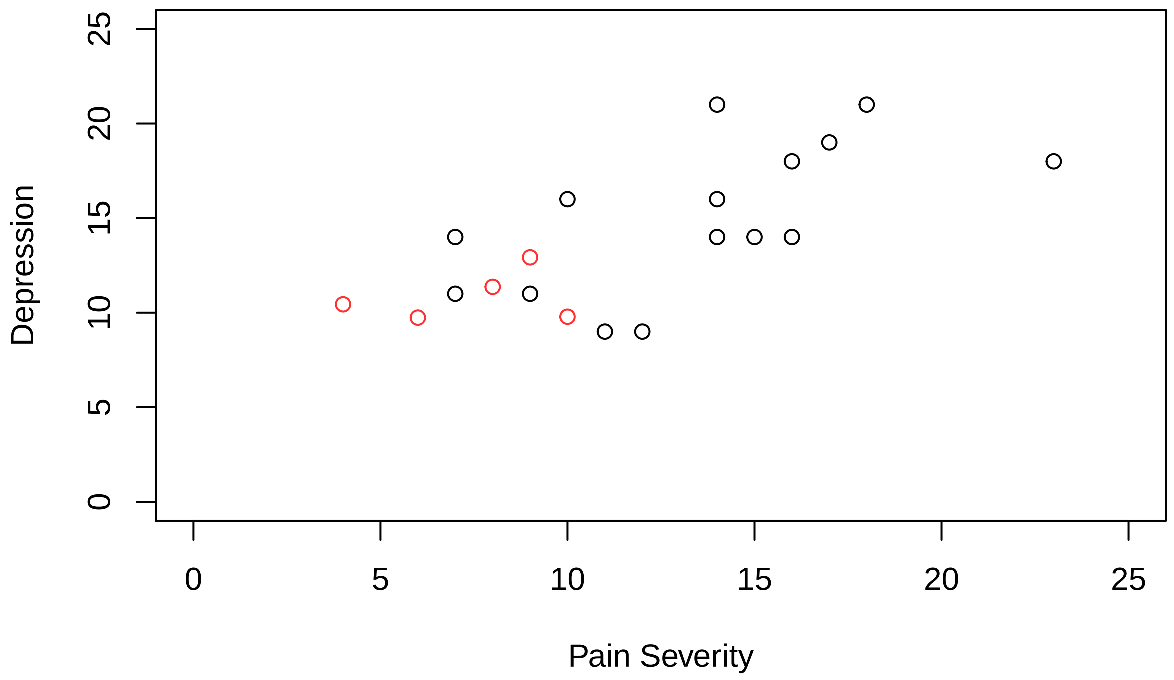
```r
qqnorm(rstandard(fit), xlim = c(-3, 3), ylim = c(-3, 3))
qqline(rstandard(fit), col = 2)
```



Finally, for stochastic regression imputation, we will add noise to the predictions obtained using the regression imputation method. Since this involves generating random numbers (from a normal distribution), it is good practice to fix the seed.

```r
set.seed(111)
predicted_sri <- predict(fit, newdata = data) + rnorm(nrow(data), 0, sigma(fit))

plot(data$PS, data$DEP, xlim = c(0, 25), ylim = c(0, 25), xlab = "Pain Severity",
    ylab = "Depression")
points(4, predicted_sri[1], col = "firebrick1")
points(6, predicted_sri[2], col = "firebrick1")
points(8, predicted_sri[5], col = "firebrick1")
points(9, predicted_sri[6], col = "firebrick1")
points(10, predicted_sri[8], col = "firebrick1")
```



```r
# completed DEP variable
DEP_sri <- ifelse(is.na(data$DEP), predicted_sri, data$DEP)

# std deviation for the observed and completed data
sd(data$DEP, na.rm = TRUE)
```

```
## [1] 3.927922
```

```r
sd(DEP_sri)
```

```
## [1] 3.890844
```

```r
# correlation for the observed and completed data
cor(data$DEP, data$PS, use = "complete")
```

```
## [1] 0.6158968
```

```r
cor(DEP_sri, data$PS)
```

```
## [1] 0.7017486
```

We can also implement the simplest version of a hot deck approach, where we randomly choose one of the individuals with an observed value for the depression score to donate their value to the individual with a missing value.

```r
# simplest hot deck possible donors (those with observed DEP scores)
ind_donors_1 <- which(is.na(data$DEP) == FALSE)
ind_donors_1
```

```
##  [1]  3  4  7  9 10 11 12 13 14 15 16 17 18 19 20
```

```r
set.seed(1)
donor1 <- sample(x = ind_donors_1, size = 1)
donor1
```

```
## [1] 14
```

```
donor2 <- sample(x = ind_donors_1, size = 1)
donor2
```

```
## [1] 9
```

```
donor5 <- sample(x = ind_donors_1, size = 1)
donor5
```

```
## [1] 12
```

```
donor6 <- sample(x = ind_donors_1, size = 1)
donor6
```

```
## [1] 3
```

```
donor8 <- sample(x = ind_donors_1, size = 1)
donor8
```

```
## [1] 4
```

```
# constructing the completed DEP variable
DEP_hotdeck <- c(data$DEP[donor1], data$DEP[donor2], data$DEP[3], data$DEP[4], data$DEP[donor5],
    data$DEP[donor6], data$DEP[7], data$DEP[donor8], data$DEP[9:20])
DEP_hotdeck
```

```
##  [1] 21 16 14 11 14 14 11 11 16  9  9 14 16 21 14 14 18 19 21 18
```

We can improve the previous procedure and decide that now the donors are those with pain scores in a similar range, which we define here, only with an illustrative purpose, by the median (so two groups: one formed by those with a pain score below the pain score median and another one formed by those with a pain score above the median of the pain score variable). Because all individuals with missing values on the depression score have their observed pain score below the median pain score, we will only be looking for donors from the first group. Below I am just illustrating how to obtain the pool of donors.

```
ps_median <- median(data$PS, na.rm = TRUE)
ps_median
```

```
## [1] 11.5
```

```
ind_donors_2 <- which(is.na(data$DEP) == FALSE & data$PS < ps_median)
ind_donors_2
```

```
## [1]  3  4  7  9 10
```

It is worth remarking that in this toy example our pool of donors is very reduced. Here it is okay because our goal is just to illustrate how the approach works but, in practice (i.e., with real data), such a reduced pool of donors is undesirable as it has the potential to distort the results (e.g., the same donor picked up several times and inherent loss of variability, etc).