Section04

```
b_ols <- function(data, y_var, X_vars, intercept = TRUE) {
  # Require the 'dplyr' package
  require(dplyr)

  # Create the y matrix
  y <- data %>%
    # Select y variable data from 'data'
    select_(.dots = y_var) %>%
    # Convert y_data to matrices
    as.matrix()

  # Create the X matrix
  X <- data %>%
    # Select X variable data from 'data'
    select_(.dots = X_vars)

  # If 'intercept' is TRUE, then add a column of ones
  # and move the column of ones to the front of the matrix
  if (intercept == T) {
    # Bind on a column of ones
    X <- cbind(1, X)
    # Name the column of ones
    names(X) <- c("ones", X_vars)
    }

  # Convert X_data to a matrix
  X <- as.matrix(X)

  # Calculate beta hat
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

  # If 'intercept' is TRUE:
  # change the name of 'ones' to 'intercept'
  if (intercept == T) rownames(beta_hat) <- c("intercept", X_vars)

  # Return beta_hat
  return(beta_hat)
}
```

```
# Load the 'dplyr' package
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

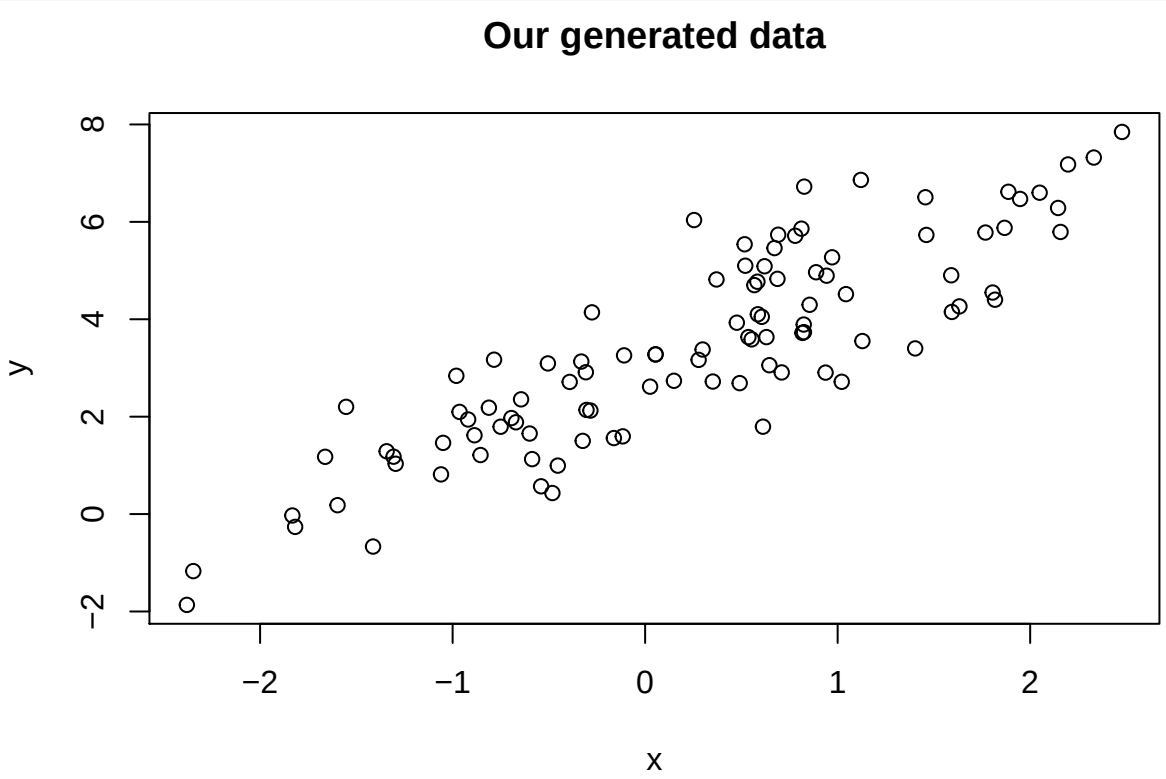
```
## The following objects are masked from 'package:base':
```

```
##
##     intersect, setdiff, setequal, union
```

```
# Set the seed
set.seed(12345)
# Set the sample size
n <- 100
# Generate the x and error data from N(0,1)
the_data <- tibble(
  x = rnorm(n),
  e = rnorm(n))
# Calculate y = 3 + 1.5 x + e
the_data <- mutate(the_data, y = 3 + 1.5 * x + e)
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.2
```

```
# Plot to make sure things are going well.
plot(
  # The variables for the plotw
  x = the_data$x, y = the_data$y,
  # Labels and title
  xlab = "x", ylab = "y", main = "Our generated data")
```




```
b_ols(data = the_data, y_var = "y", X_vars = "x", intercept = F)
```

```
##          y
## x 2.16881
```

```
library(lfe)
```

```
## Loading required package: Matrix
```

```r
# With an intercept:
felm(y ~ x, data = the_data) %>% summary()
```

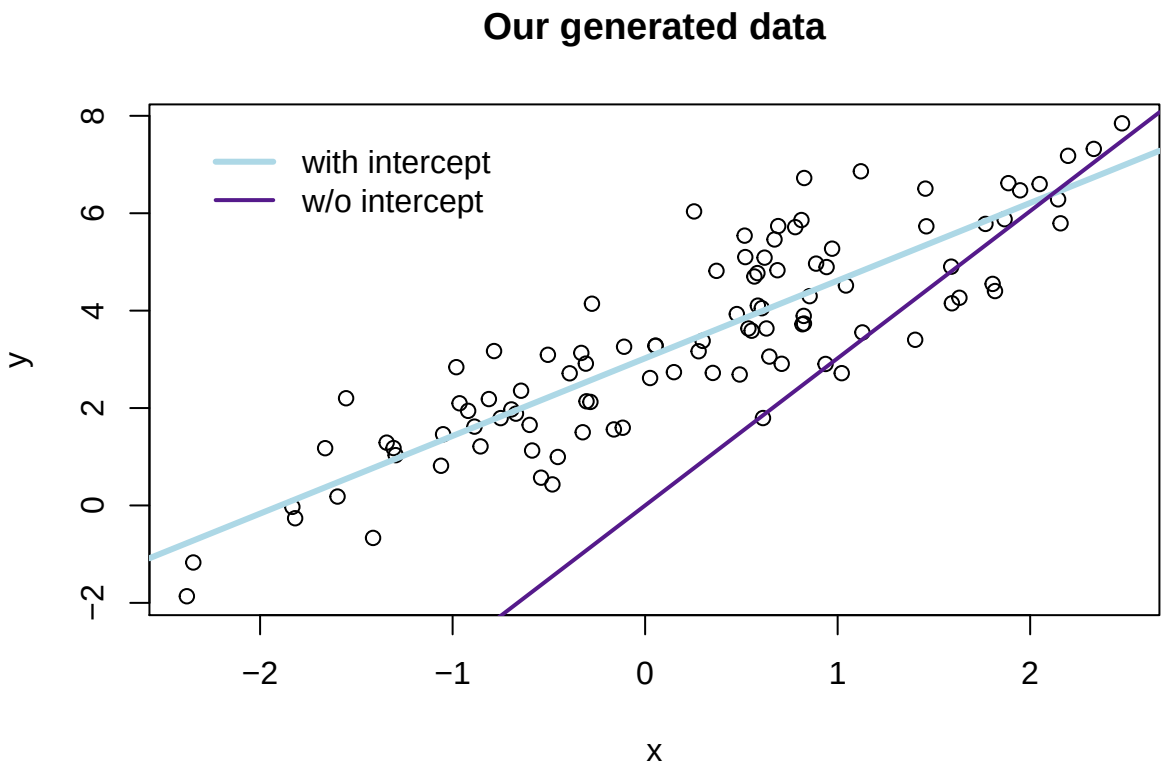
```
##
## Call:
##    felm(formula = y ~ x, data = the_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.20347 -0.60278 -0.01114  0.61898  2.60970
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.02205    0.10353   29.19   <2e-16 ***
## x            1.59454    0.09114   17.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 98 degrees of freedom
## Multiple R-squared(full model): 0.7575   Adjusted R-squared: 0.755
## Multiple R-squared(proj model): 0.7575   Adjusted R-squared: 0.755
## F-statistic(full model):306.1 on 1 and 98 DF, p-value: < 2.2e-16
## F-statistic(proj model): 306.1 on 1 and 98 DF, p-value: < 2.2e-16
```

```r
# Without an intercept:
felm(y ~ x - 1, data = the_data) %>% summary()
```

```
##
## Call:
##    felm(formula = y ~ x - 1, data = the_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## 0.3592 2.0732 2.8736 3.7755 5.5697
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x   2.1688     0.2757   7.867 4.62e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.132 on 99 degrees of freedom
## Multiple R-squared(full model): -1.351   Adjusted R-squared: -1.375
## Multiple R-squared(proj model): -1.351   Adjusted R-squared: -1.375
## F-statistic(full model):-56.89 on 1 and 99 DF, p-value: 1
## F-statistic(proj model): 61.89 on 1 and 99 DF, p-value: 4.62e-12
```

```r
# The estimates
b_w <- b_ols(data = the_data, y_var = "y", X_vars = "x", intercept = T)
b_wo <- b_ols(data = the_data, y_var = "y", X_vars = "x", intercept = F)
# Plot the points
plot(
  # The variables for the plot
  x = the_data$x, y = the_data$y,
  # Labels and title
```

```
  xlab = "x", ylab = "y", main = "Our generated data")
# Plot the line from the 'with intercept' regression in yellow
abline(a = b_w[1], b = b_w[2], col = "lightblue", lwd = 3)
# Plot the line from the 'without intercept' regression in purple
abline(a = 0, b = b_w[1], col = "purple4", lwd = 2)
# Add a legend
legend(x = min(the_data$x), y = max(the_data$y),
  legend = c("with intercept", "w/o intercept"),
  # Line widths
  lwd = c(3, 2),
  # Colors
  col = c("lightblue", "purple4"),
  # No box around the legend
  bty = "n")
```



CANNED REGRESSIONS

y ~ x y ~ x + I(x^2) #how to get a squared term y ~ x:z #just gives xz y ~ x*z #gives you x, z and xz