Statistical Analysis and Document Mining

Complementary Course: Multiple Linear Regression

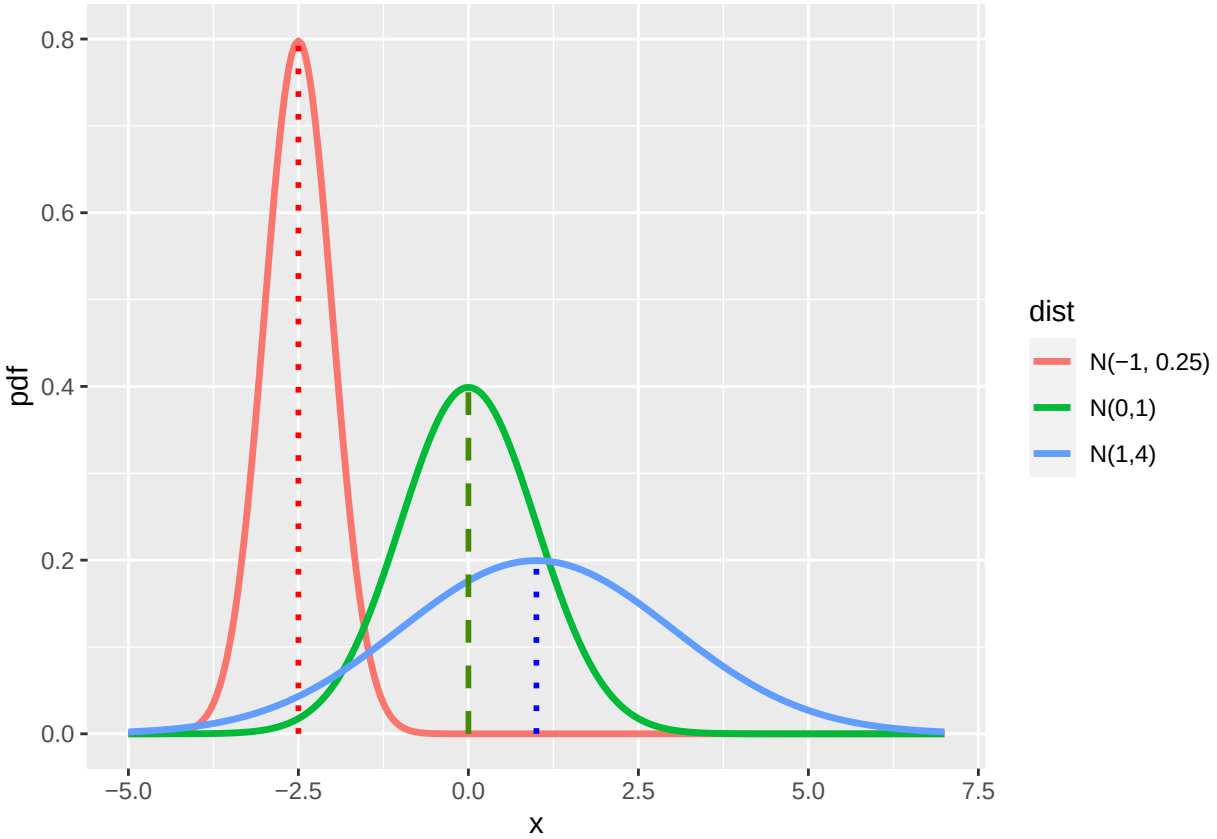
Pedro L. C. Rodrigues, TrungTin Nguyen

07/02/2023

Part 1: Multiple regression on simulated data

1) Set the seed of the random generator to 0 (set.seed(0)). Simulate $6,000 \times 201 = 1,206,000$ independent random variables with the standard normal distribution. Store them into a matrix, then into a data frame with $6,000$ lines and 201 columns. Each of these columns is referred to as a variable. Useful commands: rnorm, matrix, data.frame.

First, let's review the definition of the normal distribution, also known as the *Gaussian* distribution. If $X \sim \mathcal{N}(\mu, \sigma^2)$, we say that X is a random variable following a normal distribution with mean μ and variance σ^2. The *standard normal distribution* is a special case of the normal distribution with mean 0 and standard deviation 1.



To generate random numbers from the normal distribution, we can use rnorm(n, mean, sd), where

- The argument n is the number of random numbers we want to generate,
- The arguments mean and sd represent the mean, μ, and the standard deviation, σ, of the normal distribution, respectively.

Note that if we give only the argument n, we will generate random numbers from $\mathcal{N}(0,1)$. In other words, in R we can use rnorm(n) to generate n random numbers from the standard normal distribution $\mathcal{N}(0,1)$, which gives a similar generation to rnorm(n, mean = 0, sd = 1).

Other useful normal distribution functions can be found in the table below.

Syntax	Description	Argument
dnorm(x, mean, sd)	probability density function	x the location(s) to compute pdf
pnorm(q, mean, sd)	cumulative distribution function	q the quantile(s)
qnorm(p mean, sd)	quantile function	p the probability(es)

Solution.

```
set.seed(0)
# number of samples
n <- 6000
# number of predictors
p <- 201
# generate an array of real numbers from standard normal
M <- rnorm(n*p, mean=0, sd=1)
# create a n-by-p matrix with the sampled points
M <- matrix(M, nrow=n, ncol=p)
# create a data frame from the data matrix created above
df <- data.frame(M)
```

2) Define a Gaussian multiple linear regression model using the last 200 variables to predict the first one. In the report, write a mathematical equation (do not write R code!) to define this model. Write a mathematical equation defining the true regression model associated with the data. Compare both models.

Solution. Calling the column i of matrix M by X_i we have that the model behind the data generation is

$$X_1 = \varepsilon$$

and the multiple linear regression model we are interested in estimating can be written as

$$X_1 = \sum_{j=2}^{201} \beta_j X_j + \varepsilon$$

Ideally, we would expect that all of the β_j should be zero, since the dependent variable has been created with no dependency of the predictors whatsoever.

3) Estimate the parameters of the linear model using the last 200 variables to predict the first one. Compute the number of coefficients assessed as significantly non-zero at level 5%. Comment the result. Useful commands: summary(reg)$coefficients.

To fit a linear model in the R language, use the lm(model, dataframe) function. We can then use the summary() function to display the summary of the linear model. The summary() function interprets the most important statistical values for the analysis of the linear model.

Solution.

We calculate the coefficients of a multiple linear regression using the first column as dependent variable and all of the others as predictors

```
# terms of all the other columns in the matrix
mylm <- lm(X1 ~ ., data=df)
# note that if we run summary(mylm) we will end up with
# an output with a lot of coefficients; we don't want that
# instead, we would like to give the names of the coefficients
# whose t-statistic rejected the null hypothesis
select <- summary(mylm)$coefficients[,4] < 0.05
coeffs <- names(mylm$coefficients)[select]
print(paste('There are', length(coeffs),
            'statistically significant coefficients.'))
```

```
## [1] "There are 9 statistically significant coefficients."
```

```
print(paste('This is based on a p-value smaller than 0.05 (when there should be none)'))
```

```
## [1] "This is based on a p-value smaller than 0.05 (when there should be none)"
```

```
print(coeffs)
```

```
## [1] "X94"  "X125" "X136" "X141" "X148" "X169" "X172" "X177" "X200"
```

This is not surprising, since we have 200 predictors and are using a 0.05 threshold for the t-tests. Therefore, we could expect around $0.05 \times 200 = 10$ coefficients to be considered as different than zero.

4) Simulate a sample of size $n = 1000$ of the following model:

$$X_{1,i} = \epsilon_{1,i} \tag{1}$$
$$X_{2,i} = 3X_{1,i} + \epsilon_{2,i} \tag{2}$$
$$Y_i = X_{2,i} + X_{1,i} + 2 + \epsilon_{3,i} \tag{3}$$

where $i \in \{1, \dots, n\} =: [n]$ and the $\epsilon_{j,i}$ are independent $\mathcal{N}(0,1)$ random variables. For a given i, what is the distribution of $(X_{1,i}, X_{2,i})$? Plot the clouds of points of the simulated values of $(X_{1,i}, X_{2,i})_{i\in[n]}$. What is its shape? Why?

Solution. We first simulate the sample in R in the following.

For a given $i \in [n]$, we have that $X_{1,i} = \epsilon_{1,i} \sim \mathcal{N}(0,1)$. It follows that $3X_{1,i} \sim \mathcal{N}(0,9)$, hence $X_{2,i} = 3X_{1,i} + \epsilon_{2,i} \sim \mathcal{N}(0,10)$. By the properties of the Gaussian distribution, $(X_{1,i}, X_{2,i})$ follows a bivariate normal distribution, $\mathcal{N}(\mu, \Sigma)$ where:

- the vector mean $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$,
- the covariance matrix
$$\Sigma = \begin{pmatrix} var(X_{1,i}) & cov(X_{1,i}, X_{2,i}) \\ cov(X_{2,i}, X_{1,i}) & var(X_{2,i}) \end{pmatrix},$$
with $var(X_{1,i}) = 1$, $var(X_{1,i}) = 10$ and $cov(X_{1,i}, X_{2,i}) = cov(X_{2,i}, X_{1,i}) = 3$. Therefore, by conclusion, $(X_{1,i}, X_{2,i})$ follows a bivariate normal distribution with

$$\begin{pmatrix} X_{1,i} \\ X_{2,i} \end{pmatrix} \sim \mathcal{N}\Big(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 3 \\ 3 & 10 \end{pmatrix} \Big). \tag{4}$$

In summary, if the data points are sampled as

$$\begin{aligned} X_1 &= \varepsilon_1 \\ X_2 &= 3X_1 + \varepsilon_2 \end{aligned}$$

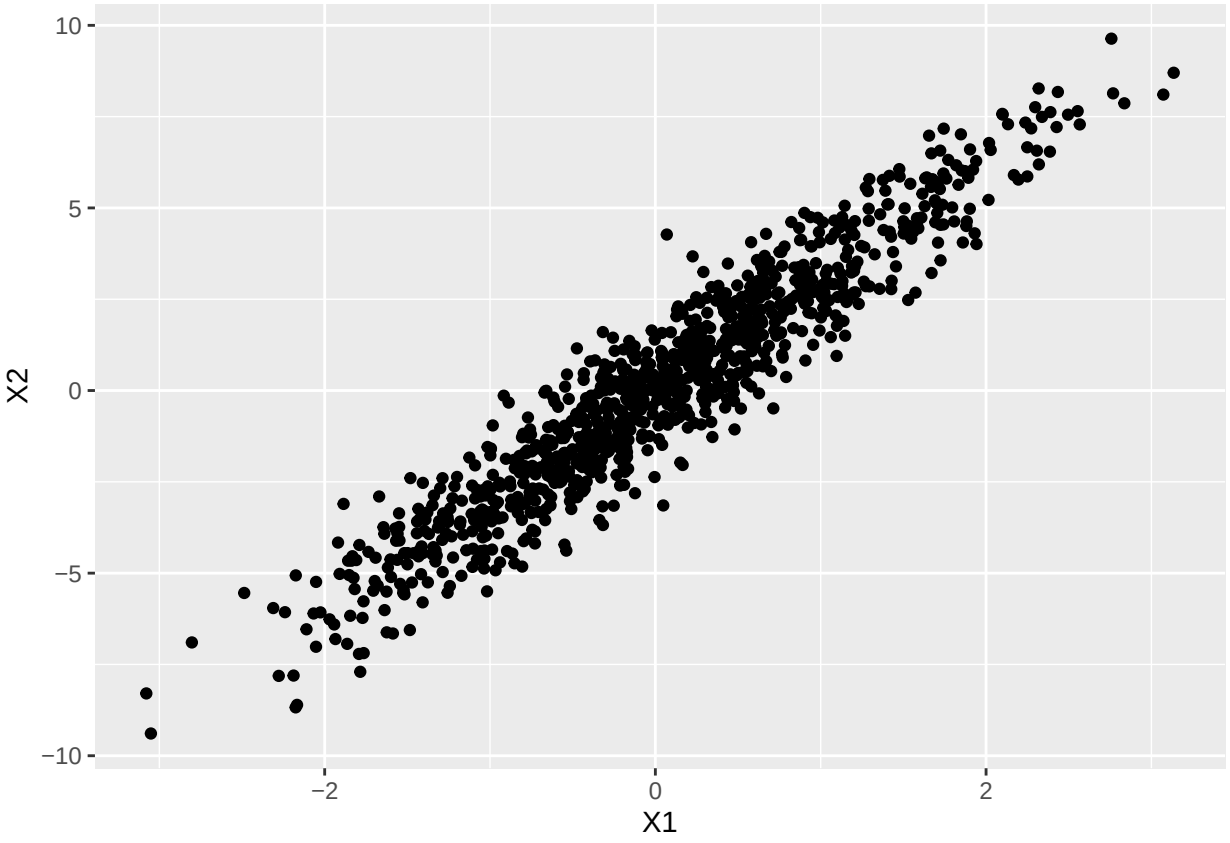
with $\varepsilon_1, \varepsilon_2$ independent and both following a standard normal distribution, then we can write the joint distribution of (X_1, X_2) as

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} \sim \mathcal{N}(\mu, C)$$

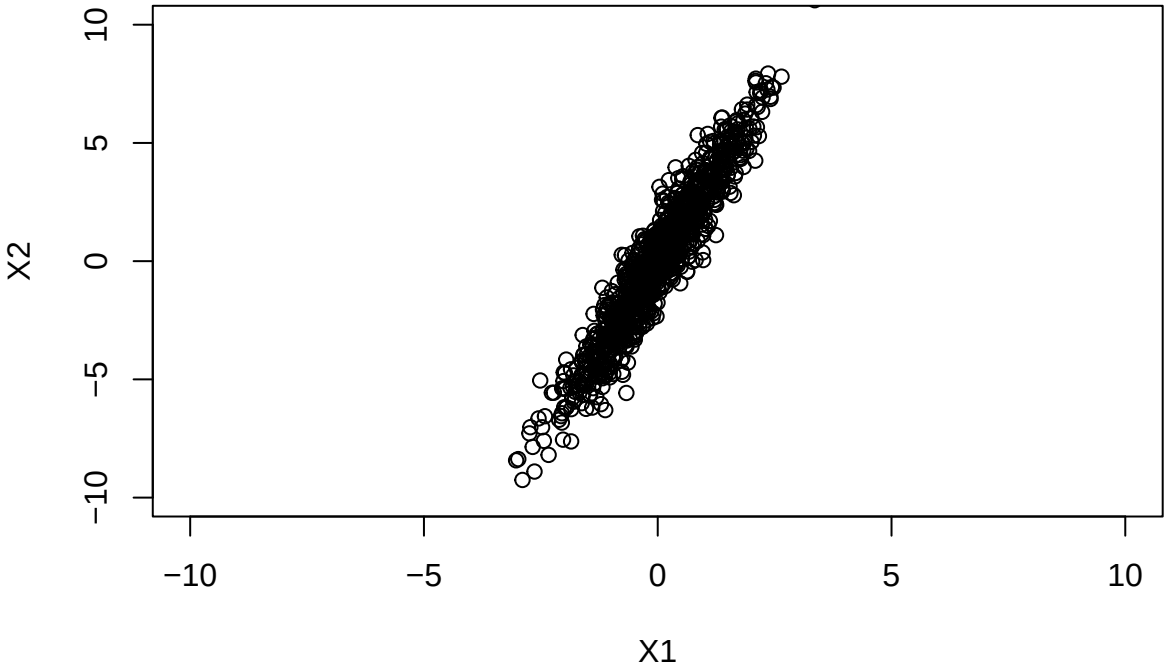
with $\mu = [0, 0]^T$ and $C = \begin{bmatrix} 1 & 3 \\ 3 & 10 \end{bmatrix}$

There are two popular ways of plotting graphs in R: using library(ggplot2) or plot().

```r
# Set the size of the sample
n <- 1000

# Sample X1
X1 <- rnorm(n = n)

# Sample X2
X2 <- 3*X1 + rnorm(n = n)

# Sample Y
Y <- X2 + X1 + 2 + rnorm(n = n)

library(ggplot2)
# Generate the data frame for plotting
plot_data <- data.frame(X1 = X1, X2 = X2)

#  Plot the clouds of points of the simulated values (X1, X2)
ggplot(data=plot_data) + geom_point(aes(x=X1, y=X2))  # Scatter plot of data.
```



```r
npoints <- 1000
eps1 <- rnorm(npoints)
eps2 <- rnorm(npoints)
eps3 <- rnorm(npoints)
X1 <- eps1
X2 <- 3*X1 + eps2
Y <- X1 + X2 + 2 + eps3
df <- data.frame(cbind(Y, X1, X2))
plot(df[c(2, 3)], xlim=c(-10, +10), ylim=c(-10, +10))
```



We see that the cloud of points is a tilted ellipse with one semi-axis much thinner than the other. The fact of being an ellipse is simply because we have a Gaussian distribution and the relation between the semi-axis is directly determined by the ratio of the diagonal elements of the covariance matrix. We can even calculate the precise angle of rotation of the ellipse based on the eigenvectors of the covariance matrix, which is ~ 73 degrees.

5) Let us consider the following 2 models:

$$\textbf{Model 1: } Y_i = \beta_1 X_{1,i} + \beta_0 + \tilde{\varepsilon}_{1,i}$$

$$\textbf{Model 2: } Y_i = \beta_2 X_{2,i} + \beta_0 + \tilde{\varepsilon}_{2,i}$$

where the $\tilde{\varepsilon}_{j,i}$ are independent $\mathcal{N}(0, \sigma^2)$ random variables. For n = 1000, check that the estimates of the parameters $\beta_0, \beta_1, \beta_2, \sigma^2$ are close to the true values. Now set the seed to 3 and simulate again $X_{1,i}, X_{2,i}, Y_i$ for $n = 10$. Estimate the parameters. What happens?

Solution.

We now consider two different linear models.

The first one is: $Y = \beta_0 + \beta_1 X_1 + \tilde{\varepsilon}_1$

Note that we have

$$\begin{aligned} Y &= X_2 + X_1 + 2 + \varepsilon_3 \\ &= (3X_1 + \varepsilon_2) + X_1 + 2 + \varepsilon_3 \\ &= 4X_1 + 2 + (\varepsilon_2 + \varepsilon_3) \end{aligned}$$

so the true values for the coefficients are: $\beta_0 = 2, \beta_1 = 4, \sigma^2 = 2$

```
lm1 <- lm(Y ~ X1, data=df)
print('Model 1')
```

```
## [1] "Model 1"
```

```
print(paste('beta_0:', lm1$coefficients[1], 'and beta_1:', lm1$coefficients[2]))
```

```
## [1] "beta_0: 2.01254274503975 and beta_1: 3.98069577577301"
```

```
print(paste('and sigma^2:', summary(lm1)$sigma**2))
```

```
## [1] "and sigma^2: 1.99318313768397"
```

The second model is: $Y = \beta_0 + \beta_2 X_2 + \tilde{\varepsilon}_2$

Note that we have

$$\begin{aligned} Y &= X_2 + X_1 + 2 + \varepsilon_3 \\ &= X_2 + (X_2 - \varepsilon_2)/3 + 2 + \varepsilon_3 \\ &= \tfrac{4}{3} X_1 + 2 + \varepsilon_3 - \tfrac{1}{3}\varepsilon_2 \end{aligned}$$

so the true coefficients are $\beta_0 = 2, \beta_1 = \frac{4}{3}, \sigma^2 = \frac{10}{9}$

```
lm2 <- lm(Y ~ X2, data=df)
print('Model 2')
```

```
## [1] "Model 2"
```

```
print(paste('beta_0:', lm2$coefficients[1], 'and beta_2:', lm2$coefficients[2]))
```

```
## [1] "beta_0: 2.02793090493895 and beta_2: 1.30059332655691"
```

```
print(paste('and sigma^2:', summary(lm2)$sigma**2))
```

```
## [1] "and sigma^2: 1.14602618344145"
```

We now resample the data with a different seed and reduce the number of data points

```
set.seed(3)

npoints <- 10
eps1 <- rnorm(npoints)
eps2 <- rnorm(npoints)
eps3 <- rnorm(npoints)
X1 <- eps1
X2 <- 3*X1 + eps2
Y <- X1 + X2 + 2 + eps3
df_ <- data.frame(cbind(Y, X1, X2))

lm1_ <- lm(Y ~ X1, data=df_)
print('Model 1 with fewer samples')
```

```
## [1] "Model 1 with fewer samples"
```

```
print(paste('beta_0:', lm1_$coefficients[1], 'and beta_1:', lm1_$coefficients[2]))
```

```
## [1] "beta_0: 1.37326832356321 and beta_1: 4.08613641830979"
```

```
print(paste('and sigma^2:', summary(lm1_)$sigma**2))
```

```
## [1] "and sigma^2: 1.05456593912006"
```

```
lm2_ <- lm(Y ~ X2, data=df_)
print('Model 2 with fewer samples')
```

```
## [1] "Model 2 with fewer samples"
```

```
print(paste('beta_0:', lm2_$coefficients[1], 'and beta_2:', lm2_$coefficients[2]))
```

```
## [1] "beta_0: 1.76624570638636 and beta_2: 1.42387656444064"
```

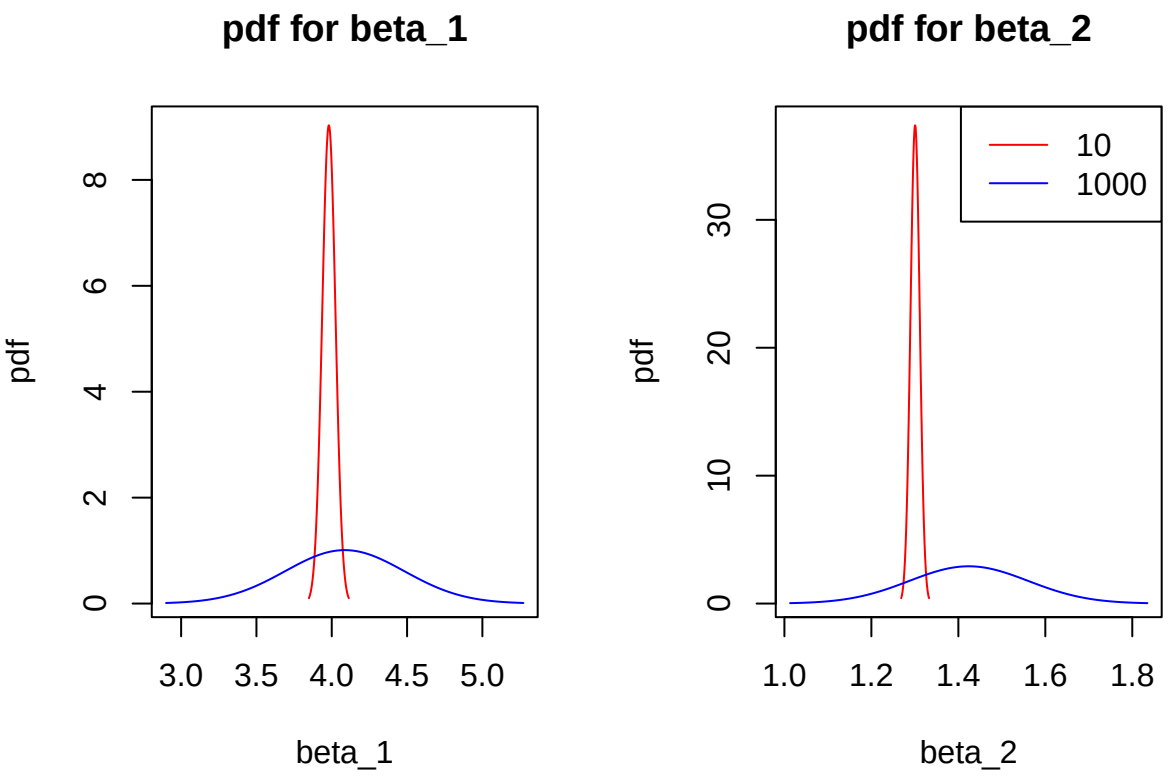
```
print(paste('and sigma^2:', summary(lm2_)$sigma**2))
```

```
## [1] "and sigma^2: 1.04282219851467"
```

We see that the estimated values with $n = 10$ are different from those for $n = 1000$.

Indeed, if we plot the theoretical distributions for the coefficients on each setting (i.e. $n = 10$ vs $n = 1000$) we have

```
par(mfrow=c(1,2))

mu1 <- summary(lm1)$coefficients[2,1]; sd1 <- summary(lm1)$coefficients[2,2]
mu1_ <- summary(lm1_)$coefficients[2,1]; sd1_ <- summary(lm1_)$coefficients[2,2]
x <- seq(mu1-3*sd1, mu1+3*sd1, length.out=1000)
x_ <- seq(mu1_-3*sd1_, mu1_+3*sd1_, length.out=1000)
plot(x, dnorm(x, mu1, sd1) , type="l", col="red", xlim=(c(mu1_-3*sd1_, mu1_+3*sd1_)),
     xlab="beta_1", ylab="pdf", main="pdf for beta_1")
lines(x_, dnorm(x_, mu1_, sd1_) , type="l", col="blue")

mu2 <- summary(lm2)$coefficients[2,1]; sd2 <- summary(lm2)$coefficients[2,2]
mu2_ <- summary(lm2_)$coefficients[2,1]; sd2_ <- summary(lm2_)$coefficients[2,2]
x <- seq(mu2-3*sd2, mu2+3*sd2, length.out=1000)
x_ <- seq(mu2_-3*sd2_, mu2_+3*sd2_, length.out=1000)
plot(x, dnorm(x, mu2, sd2) , type="l", col="red", xlim=(c(mu2_-3*sd2_, mu2_+3*sd2_)),
     xlab="beta_2", ylab="pdf", main="pdf for beta_2")
lines(x_, dnorm(x_, mu2_, sd2_) , type="l", col="blue")
legend(x="topright", legend=c("10", "1000"), col=c("red", "blue"), lty=c(1, 1))
```



6) Let us now consider the model

Let us now consider the model

$$Y_i = \beta_2 X_{2,i} + \beta_1 X_{1,i} + \beta_0 + \varepsilon_i$$

where $i \in \{1, \ldots, n\}$ and the ε_i are independent $\mathcal{N}(0, \sigma^2)$ random variables. For the previously simulated data with $n = 10$, estimate the parameters $\beta_0, \beta_1, \beta_2, \sigma^2$. What can you say about the effects of X_1 and X_2?

Solution.

We now consider the full model with only $n = 10$ samples

```
lm_full_ <- lm(Y ~ ., data=df_)
summary(lm_full_)
```

```
##
## Call:
## lm(formula = Y ~ ., data = df_)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99896 -0.66387 -0.09066  0.71179  1.30298
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.5812     0.3234   4.890  0.00177 **
## X1            2.0628     1.2974   1.590  0.15588
## X2            0.7336     0.4519   1.623  0.14855
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.9357 on 7 degrees of freedom
## Multiple R-squared:  0.9494, Adjusted R-squared:  0.9349
## F-statistic: 65.63 on 2 and 7 DF,  p-value: 2.92e-05
```

We see that although the F-test rejects the null hypothesis, the t-test for none of the coefficients is rejected. This is most likely due to the strong correlation between X_1 and X_2 and the limited number of data points involved in estimating the coefficients. Note that for $n = 1000$ our full linear model gave

```
lm_full <- lm(Y ~ ., data=df)
summary(lm_full)
```

```
##
## Call:
## lm(formula = Y ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9517 -0.7165  0.0063  0.6785  3.2189
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.02250    0.03253  62.177   <2e-16 ***
## X1           0.97667    0.10600   9.214   <2e-16 ***
## X2           1.00419    0.03376  29.743   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 997 degrees of freedom
## Multiple R-squared:  0.942,  Adjusted R-squared:  0.9419
## F-statistic:  8096 on 2 and 997 DF,  p-value: < 2.2e-16
```

where the t-test rejected their null hypothesis.

This effect can be explained by the role of n in decreasing the standard error of the parameter estimations. For larger n, the standard error is smaller and, therefore, the t-test is larger, rejecting the null hypothesis. It works in the other way around for smaller n.