Biometria II
TP Nº 2
Modelos Lineales (Regresion Lineal Simple)

Objetivos

1. Interpretación de modelos lineales con variables cuanti- y cualitativas
2. Análisis de Supuestos en modelos lineales generales
3. Ejecutar e interpretar simulaciones de modelos y distribuciones en R

Hoja de ruta

- Ejercicio guiado (script) para resolver (Problema 1)
- Ejercicio adicional sin script (Problema 2)
- Ejericio de simulación distribución normal y modelos lineales

Parte A: EJERCICIOS

Problema 1. Fitorremediación en plantas

La fitorremediación es el proceso a través del cual algunas especies de plantas que son tolerantes a los metales pesados, los acumulan en sus órganos disminuyendo así su concentración en el ambiente que habitan. Para evaluar la capacidad fitorremediadora de la especie herbácea *Thlaspi caerulescens* se realizó un experimento en el cual se pusieron a crecer plántulas de esta especie en 32 frascos con medio de cultivo Hoagland a los que se le agregaron al azar distintas concentraciones de cadmio (de 0 a 500 μM). Luego de 14 días se cosecharon las plantas y se midió la concentración de Cd acumulada en tallos [mg.kg $^{-1}$)], obteniéndose los resultados que se encuentran en el archivo *cadmio.txt*. Para este experimento:

1) Indique la cantidad de replicas y el tipo de variables involucradas.

```
wd <- getwd()
cadmio = read.table("/home/jose/Documents/materias/biome2/2020/tps/tp2/clase/Cadmio.txt",
    header = T)
# opcional:
attach(cadmio)
```

```
# inspeccion del data.frame
class(cadmio)
```

```
## [1] "data.frame"
```

```
str(cadmio)
```

```
## 'data.frame':    32 obs. of  2 variables:
##  $ Cd        : int  5 5 5 5 10 10 10 10 25 25 ...
##  $ Cdenplanta: int  430 400 420 410 415 420 440 430 450 440 ...
```

```
dim(cadmio)
```

```
## [1] 32  2
```

```r
head(cadmio)
```

```
##   Cd Cdenplanta
## 1  5        430
## 2  5        400
## 3  5        420
## 4  5        410
## 5 10        415
## 6 10        420
```

```r
tail(cadmio)
```

```
##     Cd Cdenplanta
## 27 300        620
## 28 300        630
## 29 500        760
## 30 500        730
## 31 500        780
## 32 500        740
```

```r
# cantidad de replicas
table(cadmio$Cd)
```

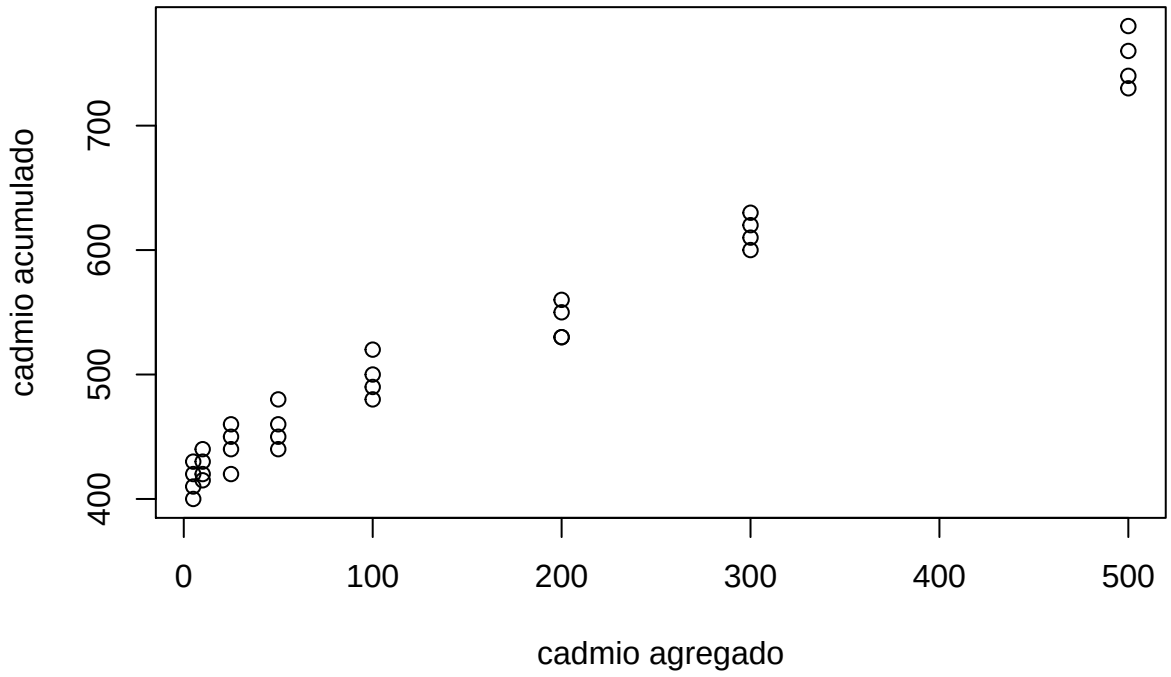
```
##
##   5  10  25  50 100 200 300 500
##   4   4   4   4   4   4   4   4
```

```r
# Una manera alternativa de hacerlo
res = c()
for (i in 1:length(levels(as.factor(cadmio$Cd)))) {
    res = c(res, nrow(cadmio[cadmio$Cd == levels(as.factor(cadmio$Cd)[i]),
        ]))
}
res
```

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

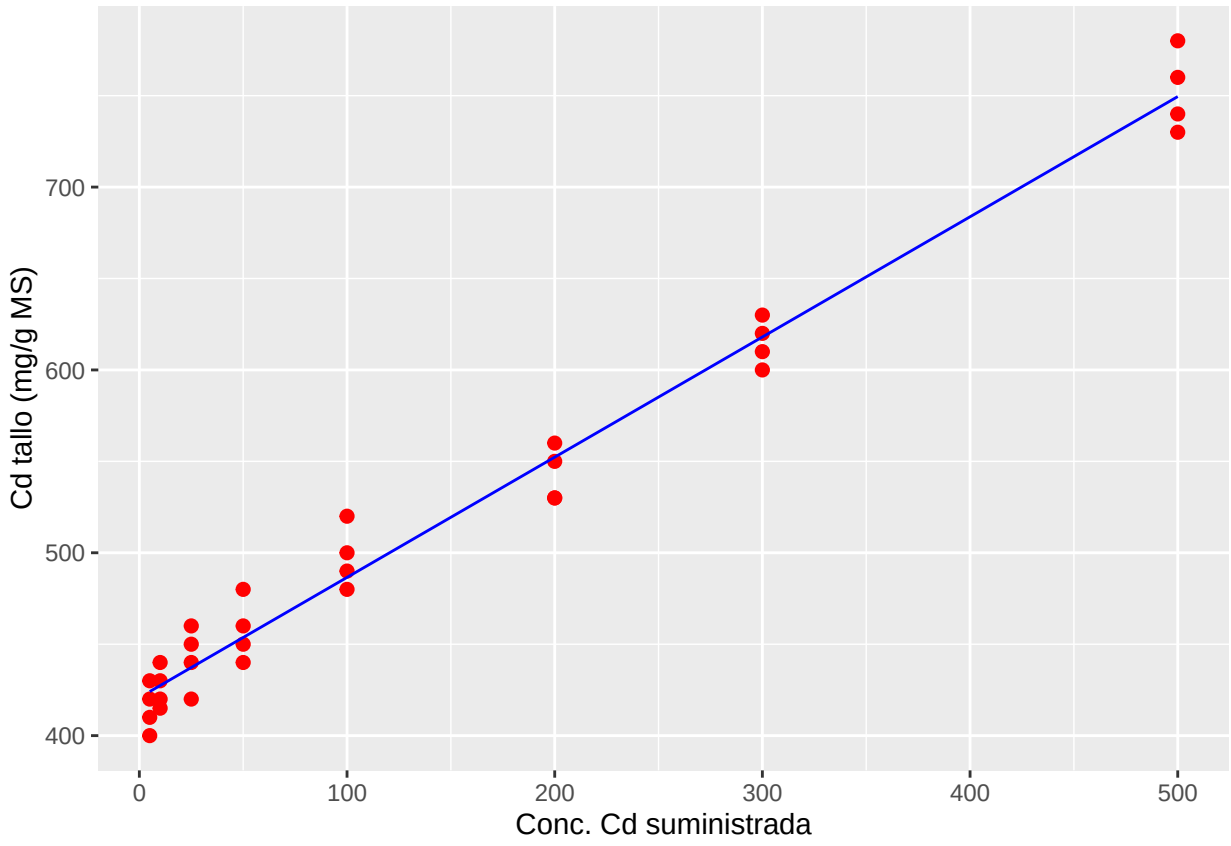
2) Describa grafica y estadisticamente los datos. (*Ayuda: tapply, summary y stat.esc)*

```r
# graficos R base
plot(cadmio, ylab = "cadmio acumulado", xlab = "cadmio agregado")

# ggplot2b
library(ggplot2)
```




```
disp <- ggplot(cadmio, aes(x = Cd, y = Cdenplanta)) + geom_point(size = 2,
    color = "red", shape = 19) + geom_smooth(method = lm, se = F,
    fullrange = F, size = 0.5, color = "blue") + xlab("Conc. Cd suministrada") +
    ylab("Cd tallo (mg/g MS)")
disp
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# resumen descriptiva
summary(cadmio)
```

```
##        Cd           Cdenplanta
##  Min.   :  5.00   Min.   :400.0
##  1st Qu.: 21.25   1st Qu.:437.5
##  Median : 75.00   Median :480.0
##  Mean   :148.75   Mean   :518.6
##  3rd Qu.:225.00   3rd Qu.:570.0
##  Max.   :500.00   Max.   :780.0
```

¿Que hace la funcion *tapply*?
tapply toma como argumentos (X,INDEX,FUN)
-La funcion *tapply* aplica (de ahi parte de su nombre) una funcion (FUN) a un vector (X) agrupado para cada factor definido en INDEX. En este caso, queremos que para cada valor de la categoria "Cd", aplique *summary* sobre el vector "Cdenplanta".

```
library(pastecs)
tapply(Cdenplanta, Cd, summary)
```

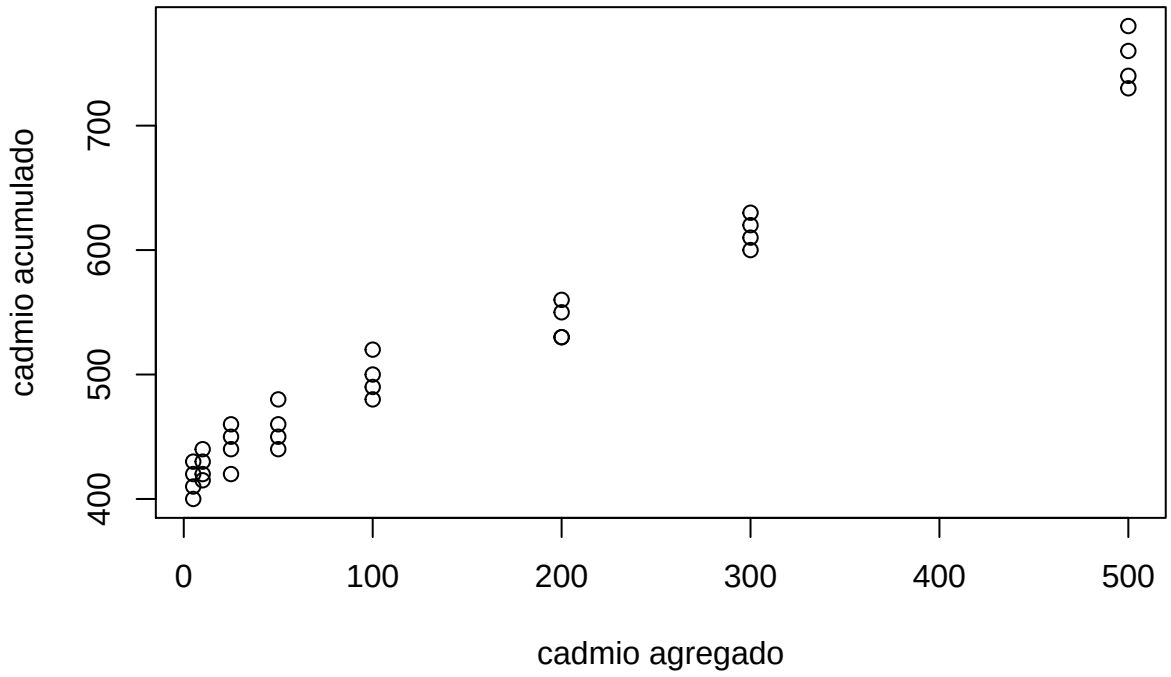
```
## $`5`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   400.0   407.5   415.0   415.0   422.5   430.0
##
## $`10`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   415.0   418.8   425.0   426.2   432.5   440.0
##
## $`25`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   420.0   435.0   445.0   442.5   452.5   460.0
##
## $`50`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   440.0   447.5   455.0   457.5   465.0   480.0
##
## $`100`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   480.0   487.5   495.0   497.5   505.0   520.0
##
## $`200`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   530.0   530.0   540.0   542.5   552.5   560.0
##
## $`300`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   600.0   607.5   615.0   615.0   622.5   630.0
##
## $`500`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   730.0   737.5   750.0   752.5   765.0   780.0
```

```
tapply(Cdenplanta, Cd, stat.desc)
```

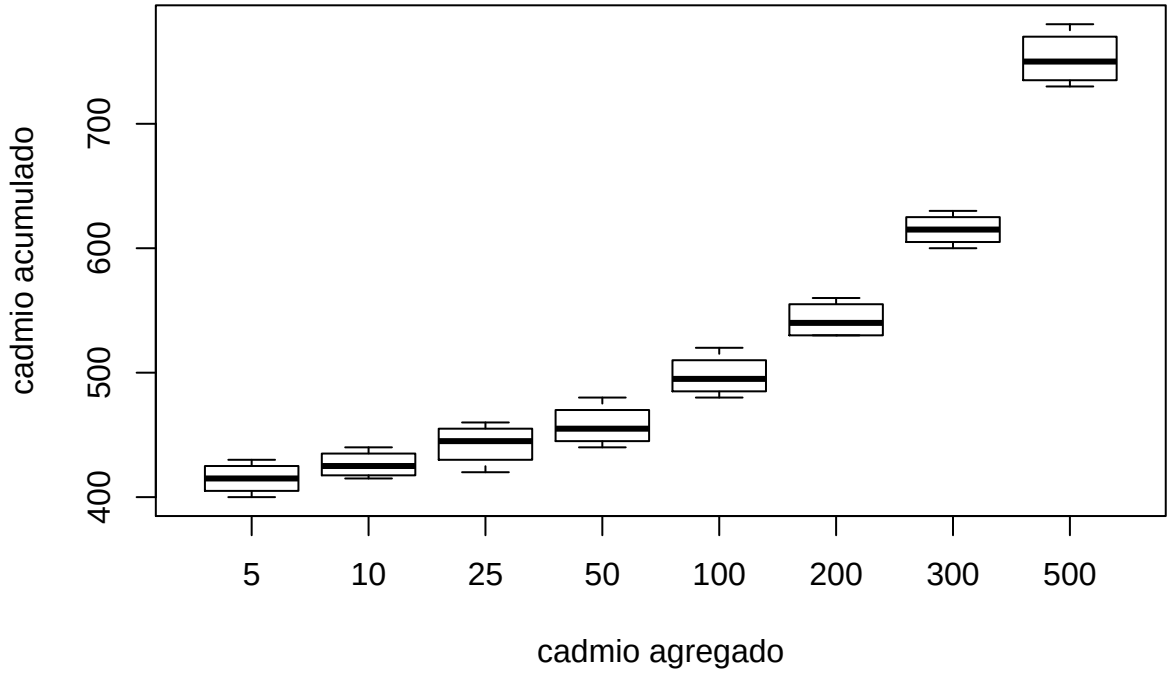
```
## $`5`
##      nbr.val     nbr.null       nbr.na          min          max        range
##    4.0000000    0.0000000    0.0000000  400.0000000  430.0000000   30.0000000
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 1660.0000000  415.0000000  415.0000000    6.4549722   20.5426026  166.6666667
##      std.dev     coef.var
##   12.9099445    0.0311083
##
## $`10`
##      nbr.val     nbr.null       nbr.na          min          max        range
```

```
## 4.000000e+00 0.000000e+00 0.000000e+00 4.150000e+02 4.400000e+02 2.500000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 1.705000e+03 4.250000e+02 4.262500e+02 5.543389e+00 1.764154e+01 1.229167e+02
##      std.dev     coef.var
## 1.108678e+01 2.601004e-02
##
## $`25`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 4.200000e+02 4.600000e+02 4.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 1.770000e+03 4.450000e+02 4.425000e+02 8.539126e+00 2.717531e+01 2.916667e+02
##      std.dev     coef.var
## 1.707825e+01 3.859492e-02
##
## $`50`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 4.400000e+02 4.800000e+02 4.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 1.830000e+03 4.550000e+02 4.575000e+02 8.539126e+00 2.717531e+01 2.916667e+02
##      std.dev     coef.var
## 1.707825e+01 3.732951e-02
##
## $`100`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 4.800000e+02 5.200000e+02 4.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 1.990000e+03 4.950000e+02 4.975000e+02 8.539126e+00 2.717531e+01 2.916667e+02
##      std.dev     coef.var
## 1.707825e+01 3.432814e-02
##
## $`200`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 5.300000e+02 5.600000e+02 3.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 2.170000e+03 5.400000e+02 5.425000e+02 7.500000e+00 2.386835e+01 2.250000e+02
##      std.dev     coef.var
## 1.500000e+01 2.764977e-02
##
## $`300`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 6.000000e+02 6.300000e+02 3.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 2.460000e+03 6.150000e+02 6.150000e+02 6.454972e+00 2.054260e+01 1.666667e+02
##      std.dev     coef.var
## 1.290994e+01 2.099178e-02
##
## $`500`
##      nbr.val     nbr.null       nbr.na          min          max        range
## 4.000000e+00 0.000000e+00 0.000000e+00 7.300000e+02 7.800000e+02 5.000000e+01
##          sum       median         mean      SE.mean CI.mean.0.95          var
## 3.010000e+03 7.500000e+02 7.525000e+02 1.108678e+01 3.528308e+01 4.916667e+02
##      std.dev     coef.var
## 2.217356e+01 2.946652e-02
```

```
plot(cadmio, ylab = "cadmio acumulado", xlab = "cadmio agregado")
```




```
plot(as.factor(Cd), Cdenplanta, ylab = "cadmio acumulado", xlab = "cadmio agregado")
```



3) Analice como se modifica la concentracion de cadmio absorbida por las plantas en relacion a la concentracion de cadmio ambiental. Plantee el modelo, compruebe los supuestos.

$$[CdAcumTallos]_{(mg/kg)i} = \beta_0 + \beta_1 * [CdSuministrada]_{(mg/kg*\mu M)i}| + \epsilon_i$$
$$\epsilon_i \sim N(0, \sigma)$$
$$i = 1 : n(32)$$

```
Modelo_Cdcuant <- lm(Cdenplanta ~ Cd, cadmio)
```

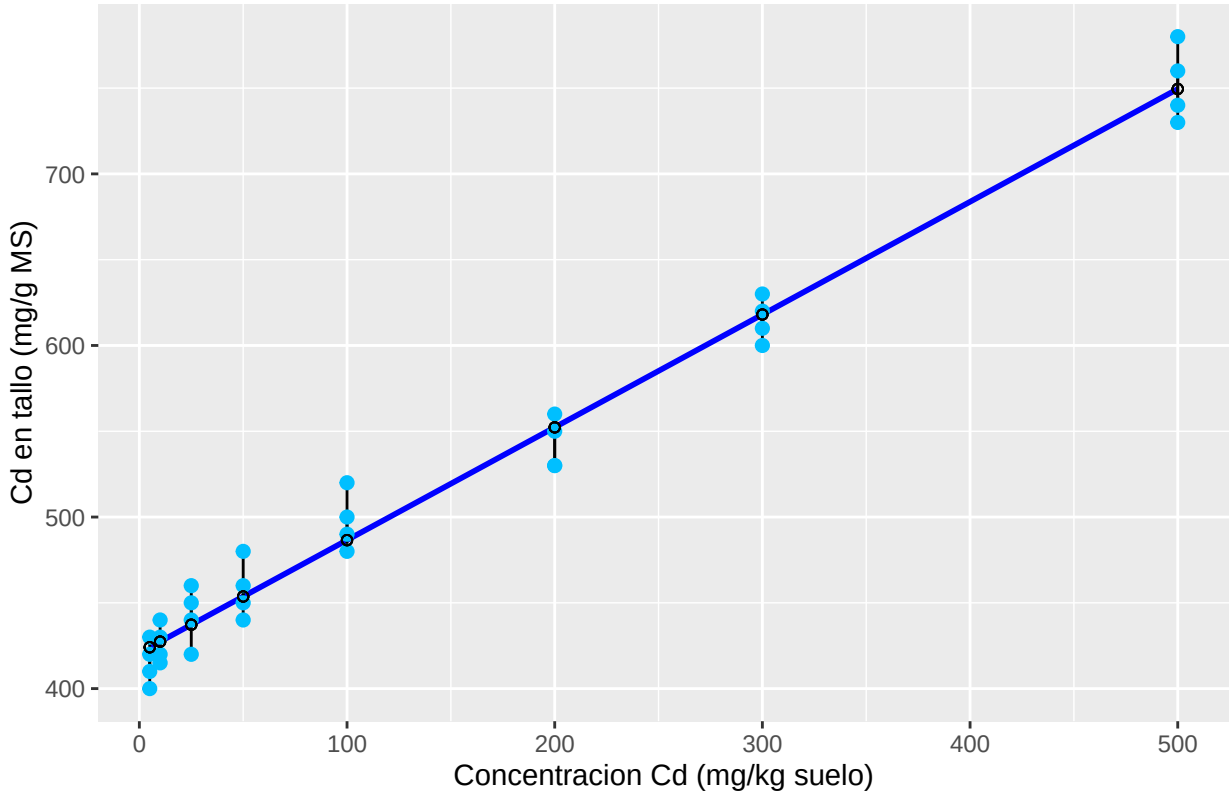
```
# Calculamos los residuos y los predichos
e <- residuals(Modelo_Cdcuant)  # residuos
re <- rstandard(Modelo_Cdcuant)  #residuos estandarizados
pre <- predict(Modelo_Cdcuant)  #predichos
```

```r
res <- cbind(cadmio$Cd, cadmio$Cdenplanta, pre, e, round(re, 2))
colnames(res) <- c("dosis Cd", "Cd tallo", "Predichos", "Residuos",
    "residuos std")
head(res)
```

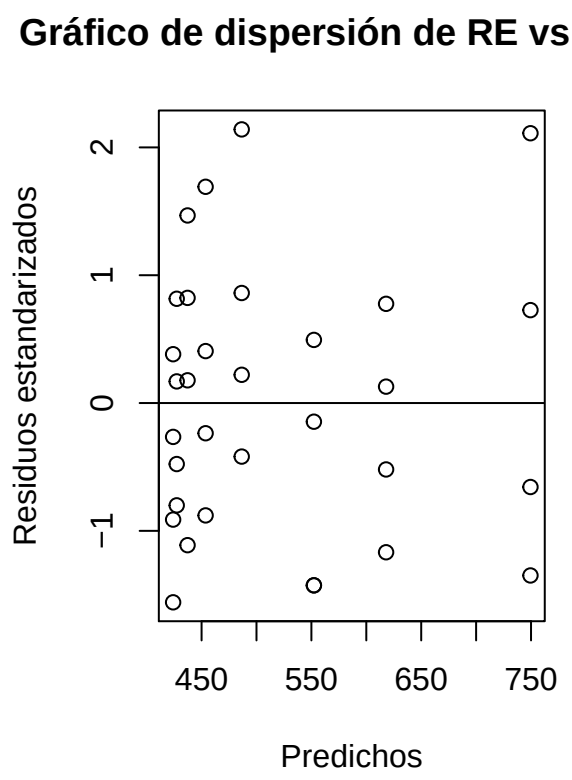
```
##   dosis Cd Cd tallo Predichos   Residuos residuos std
## 1        5      430  424.0966   5.903448         0.38
## 2        5      400  424.0966 -24.096552        -1.56
## 3        5      420  424.0966  -4.096552        -0.27
## 4        5      410  424.0966 -14.096552        -0.91
## 5       10      415  427.3834 -12.383411        -0.80
## 6       10      420  427.3834  -7.383411        -0.48
```

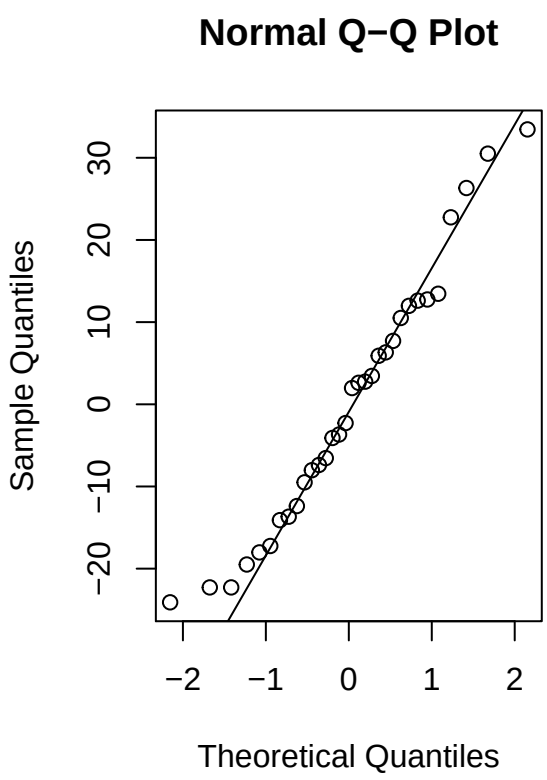
```r
# Graficamos residuos en el scatter plot
ggplot(cadmio, aes(x = Cd, y = Cdenplanta)) + geom_smooth(method = "lm",
    se = FALSE, color = "blue") + geom_segment(aes(xend = Cd, yend = pre)) +
    geom_point(aes(), colour = "deepskyblue", size = 2) + geom_point(aes(y = pre),
    shape = 1) + xlab("Concentracion Cd (mg/kg suelo)") + ylab("Cd en tallo (mg/g MS)") +
    ggtitle("")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

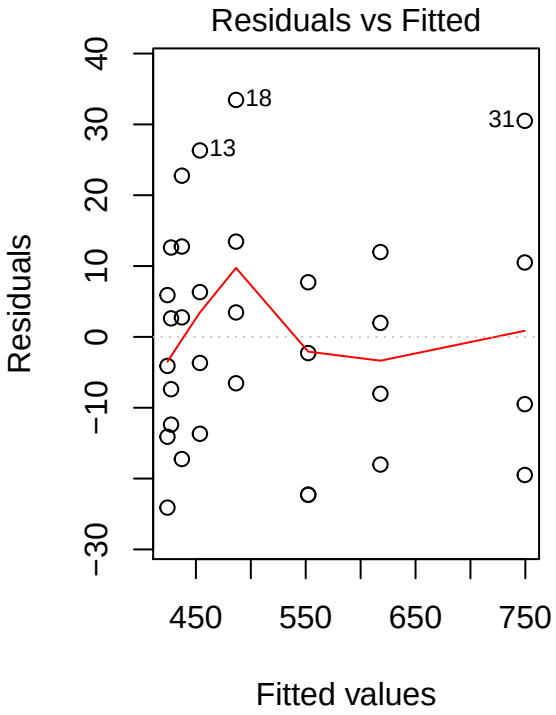


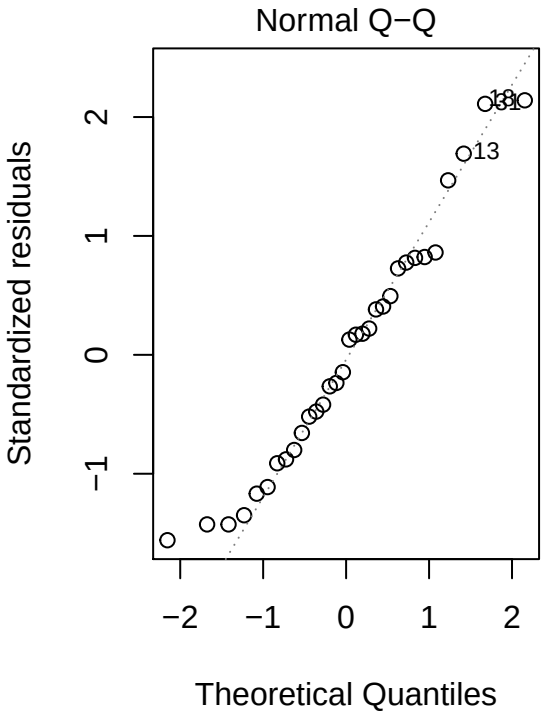

```r
# Supuestos modelo
res <- as.data.frame(res)
par(mfrow = c(1, 2))
plot(pre, re, xlab = "Predichos", ylab = "Residuos estandarizados",
    main = "Gráfico de dispersión de RE vs PRED")
abline(0, 0)
qqnorm(e)
qqline(e)
```






```
par(mfrow = c(1, 2))
plot(Modelo_Cdcuant, which = c(1, 2))
```






```
par(mfrow = c(1, 1))
shapiro.test(e)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  e
## W = 0.96473, p-value = 0.3679
```

4) ¿Considera que se puede proponer a *Thlaspi caerulescens* como un agente fitorremediador?. Justifique.

```
summary(Modelo_Cdcuant)
```

```
##
## Call:
## lm(formula = Cdenplanta ~ Cd, data = cadmio)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.097 -12.707  -0.153  10.873  33.453
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 420.8097     3.7904  111.02   <2e-16 ***
## Cd            0.6574     0.0171   38.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.9 on 30 degrees of freedom
## Multiple R-squared:  0.9801, Adjusted R-squared:  0.9794
## F-statistic:  1479 on 1 and 30 DF,  p-value: < 2.2e-16
```

```
confint(Modelo_Cdcuant)
```

```
##                   2.5 %      97.5 %
## (Intercept) 413.0686009 428.5507846
## Cd            0.6224567   0.6922869
```

Si, porque por cada μM de concentracion de cadmio que se suministra se espera que el aumento en el cadmio acumulado por tallo se encuentre entre 0,62 y 0,69 $_{mg/kg}$ de cadmio con una confianza del 95%.

5) Analice como se modifica la concentracion de cadmio absorbida por las plantas en relación a la concentración de cadmio ambiental, pero ahora considerando al cadmio ambiental como una **variable categorica**. Plantee el nuevo modelo, compruebe los supuestos e interprete los resultados obtenidos.

$$[CdAcumTallos]_{(mg/kg)i} = \beta_0 + \beta_1 * [10]_{(mg/kg)} + \beta_2 * [25]_{(mg/kg)} + \beta_3 * [50]_{(mg/kg)} + \beta_4 * [100]_{(mg/kg)} + \beta_5 * [200]_{(mg/kg)}$$

$$+\beta_6 * [300]_{(mg/kg)} + \beta_7 * [500]_{(mg/kg)} + \epsilon_i$$

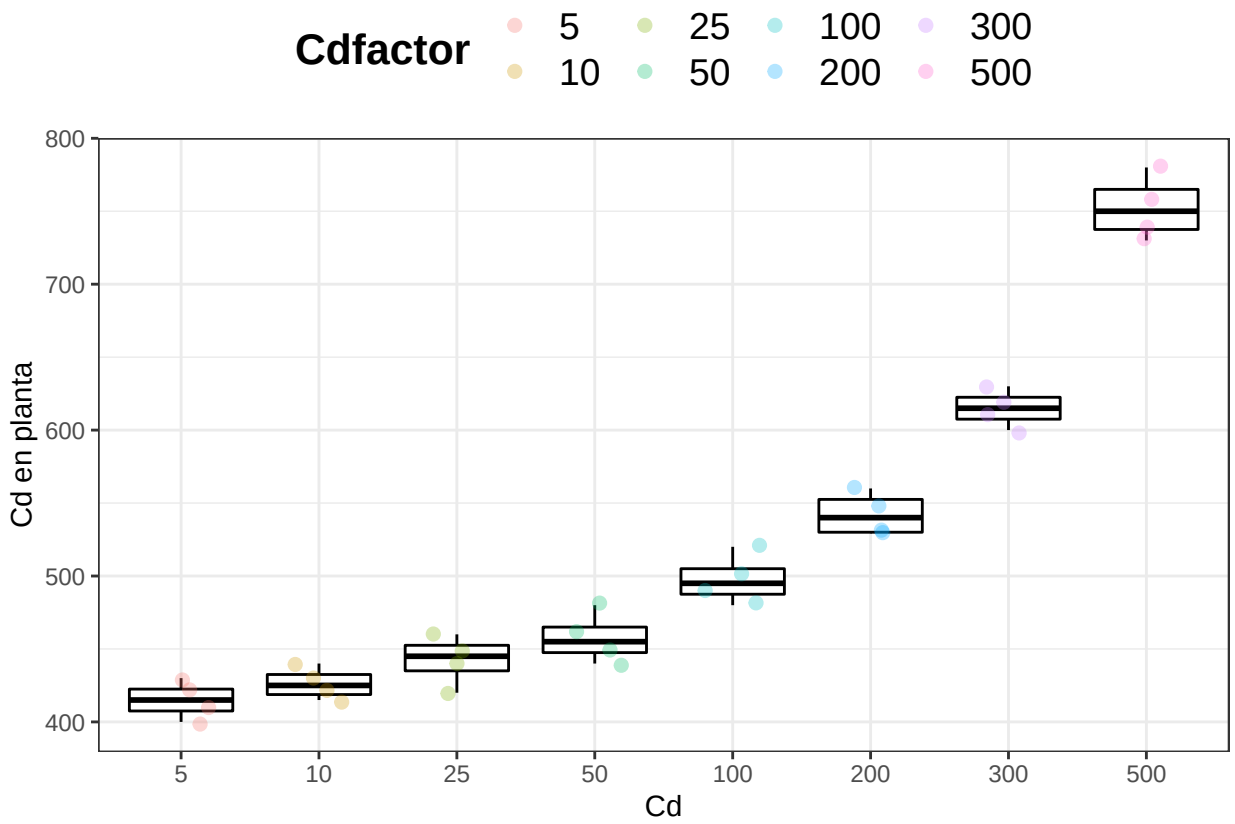
$$\epsilon_i \sim N(0, \sigma)$$

$$i = 1 : n(32)$$

```
# Pasamos la variable explicatoria (continua) a una variable
# categórica
Cdfactor <- as.factor(Cd)  #crean el objeto aparte... tambien se puede aplicar la funcion as.factor
# en el modelo
levels(Cdfactor)  # devuelve los niveles del factor
```

```
## [1] "5"   "10"  "25"  "50"  "100" "200" "300" "500"
```

```
# graficos exploratorios
box <- ggplot(cadmio, aes(x = Cdfactor, y = Cdenplanta)) + geom_boxplot(aes(color = Cdfactor),
    color = "black") + theme_bw() + geom_jitter(alpha = 0.3, size = 2,
    aes(color = Cdfactor), position = position_jitter(width = 0.2)) +
    theme(legend.position = "top", legend.text = element_text(size = 14),
        legend.title = element_text(size = 16, face = "bold")) + ylab("Cd en planta") +
    xlab("Cd")
box
```



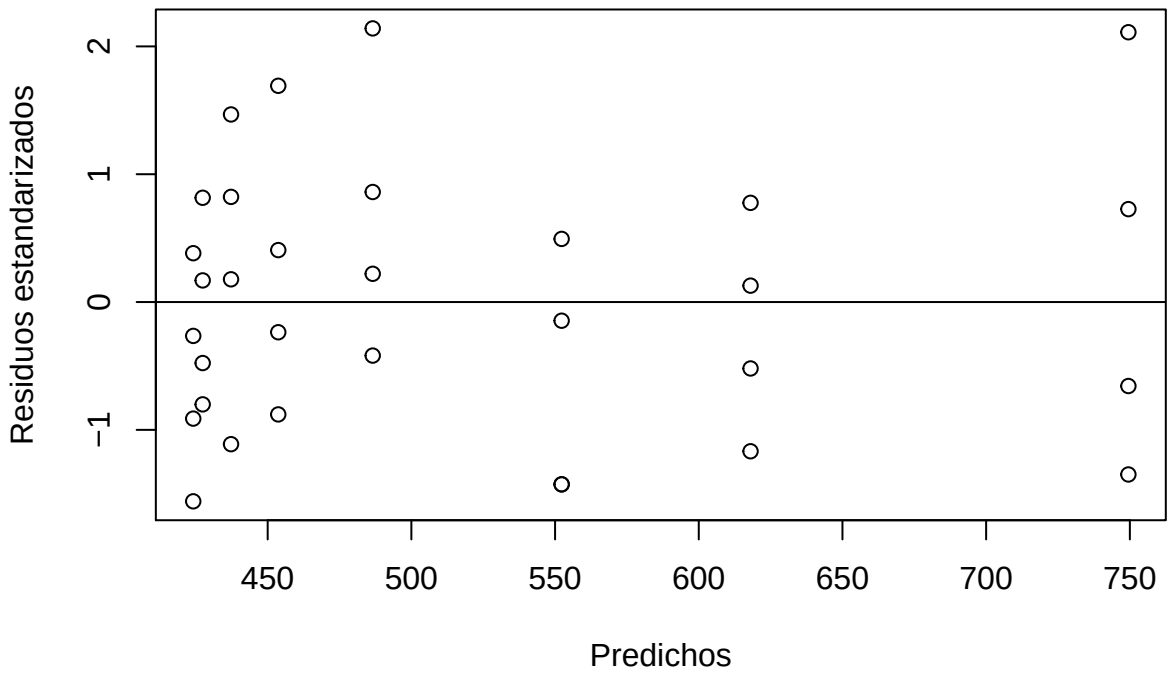

```
Modelo_CdFactor <- lm(Cdenplanta ~ Cdfactor, cadmio)
# Supuestos residuos
eFac <- residuals.lm(Modelo_CdFactor)
reFac <- rstandard(Modelo_CdFactor)  #residuos estandarizados
preFac <- predict(Modelo_CdFactor)  #predichos
resFac <- cbind(cadmio$Cd, cadmio$Cdenplanta, eFac, reFac, preFac)
colnames(resFac) <- c("Dosis Cd", "Cd tallo", "Residuos", "Residuos Std",
    "Predichos")
head(resFac)
```

```
##   Dosis Cd Cd tallo Residuos Residuos Std Predichos
## 1        5      430    15.00    1.0825538    415.00
## 2        5      400   -15.00   -1.0825538    415.00
## 3        5      420     5.00    0.3608513    415.00
## 4        5      410    -5.00   -0.3608513    415.00
## 5       10      415   -11.25   -0.8119153    426.25
## 6       10      420    -6.25   -0.4510641    426.25
```

```
shapiro.test(eFac)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  eFac
## W = 0.96035, p-value = 0.2808
```

```
plot(pre, re, xlab = "Predichos", ylab = "Residuos estandarizados",
    main = "Gráfico de dispersión de RE vs PRED")
abline(0, 0)
```



• Homogeneidad de varianza. Prueba de Levene

¿Por qué antes no pudimos hacer esta prueba?
Porque esta prueba se utiliza cuando tenemos predictoras categoricas.

```
library(car)
```

```
## Loading required package: carData
```

```
leveneTest(Modelo_CdFactor)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  7  0.4731 0.8444
##       24
```

• Vemos el resumen del modelo

```
summary(Modelo_CdFactor)
```

```
##
## Call:
## lm(formula = Cdenplanta ~ Cdfactor, data = cadmio)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.500 -12.500   0.000   9.062  27.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   415.00       8.00  51.876  < 2e-16 ***
## Cdfactor10     11.25      11.31   0.994 0.329954
## Cdfactor25     27.50      11.31   2.431 0.022911 *
## Cdfactor50     42.50      11.31   3.757 0.000972 ***
```

```
## Cdfactor100    82.50      11.31   7.292 1.56e-07 ***
## Cdfactor200   127.50      11.31  11.270 4.53e-11 ***
## Cdfactor300   200.00      11.31  17.678 2.89e-15 ***
## Cdfactor500   337.50      11.31  29.832  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16 on 24 degrees of freedom
## Multiple R-squared:  0.9839, Adjusted R-squared:  0.9792
## F-statistic: 209.4 on 7 and 24 DF,  p-value: < 2.2e-16
```

• Magnitud del Efecto

¿Cómo se evalúa la magnitud del efecto cuando la variable es un factor? *Comparaciones múltiples a posteriori. Por ej. Test de Tukey*

Investigue en google qué funciones puede utilizar para realizarlo.

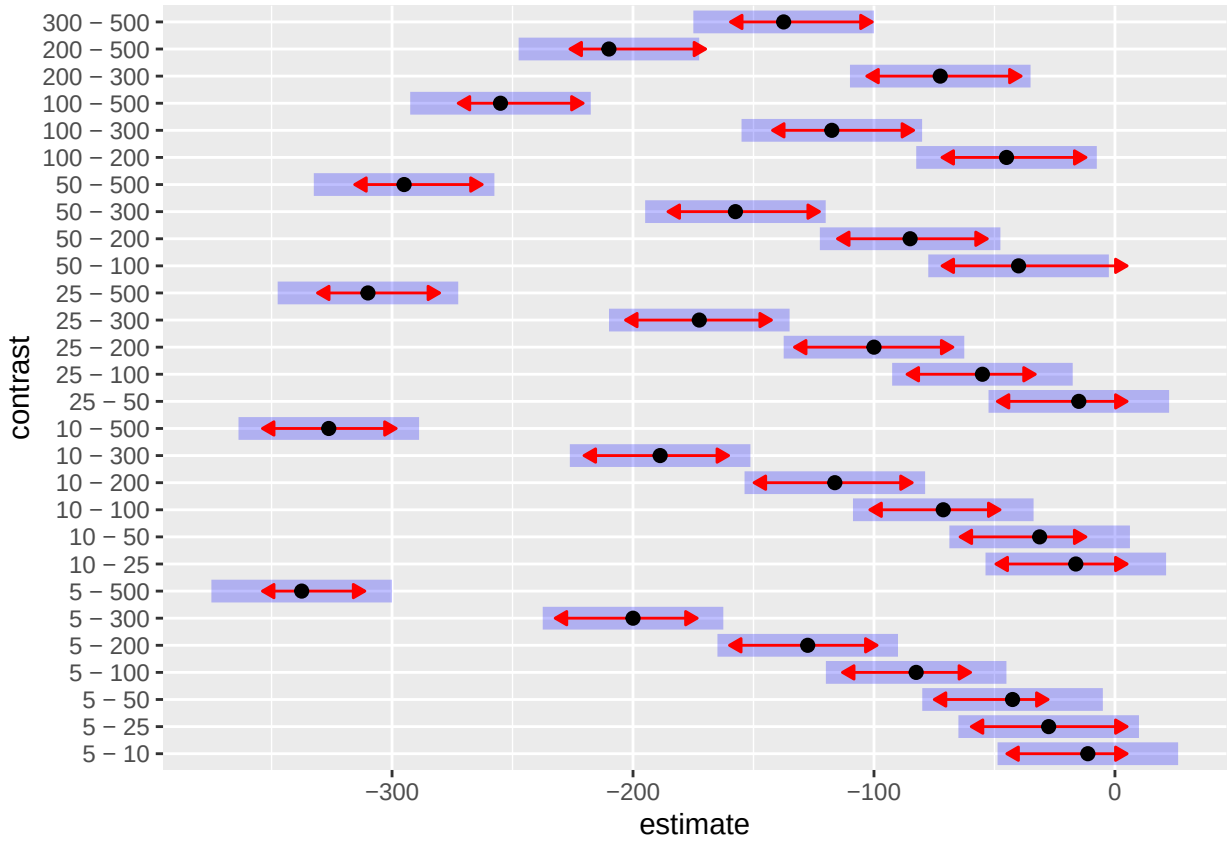
```
# Hacemos la comparacion con emmeans que es como lo vamos a hacer
# en la materia.
library(emmeans)
options(emmeans = list(emmeans = list(infer = c(FALSE, TRUE)), contrast = list(infer = c(TRUE,
    FALSE))))
TukeyEmmeans <- emmeans(Modelo_CdFactor, pairwise ~ Cdfactor)
TukeyEmmeans
```

```
## $emmeans
##  Cdfactor emmean SE df t.ratio p.value
##  5           415  8 24 51.876  <.0001
##  10          426  8 24 53.282  <.0001
##  25          442  8 24 55.314  <.0001
##  50          458  8 24 57.189  <.0001
##  100         498  8 24 62.189  <.0001
##  200         542  8 24 67.814  <.0001
##  300         615  8 24 76.877  <.0001
##  500         752  8 24 94.064  <.0001
##
##
## $contrasts
##  contrast  estimate   SE df lower.CL upper.CL
##  5 - 10       -11.2 11.3 24    -48.7    26.22
##  5 - 25       -27.5 11.3 24    -65.0     9.97
##  5 - 50       -42.5 11.3 24    -80.0    -5.03
##  5 - 100      -82.5 11.3 24   -120.0   -45.03
##  5 - 200     -127.5 11.3 24   -165.0   -90.03
##  5 - 300     -200.0 11.3 24   -237.5  -162.53
##  5 - 500     -337.5 11.3 24   -375.0  -300.03
##  10 - 25      -16.2 11.3 24    -53.7    21.22
##  10 - 50      -31.2 11.3 24    -68.7     6.22
##  10 - 100     -71.2 11.3 24   -108.7   -33.78
##  10 - 200    -116.2 11.3 24   -153.7   -78.78
##  10 - 300    -188.8 11.3 24   -226.2  -151.28
##  10 - 500    -326.2 11.3 24   -363.7  -288.78
##  25 - 50      -15.0 11.3 24    -52.5    22.47
##  25 - 100     -55.0 11.3 24    -92.5   -17.53
##  25 - 200    -100.0 11.3 24   -137.5   -62.53
##  25 - 300    -172.5 11.3 24   -210.0  -135.03
##  25 - 500    -310.0 11.3 24   -347.5  -272.53
##  50 - 100     -40.0 11.3 24    -77.5    -2.53
```

```
##  50 - 200     -85.0 11.3 24   -122.5   -47.53
##  50 - 300    -157.5 11.3 24   -195.0  -120.03
##  50 - 500    -295.0 11.3 24   -332.5  -257.53
##  100 - 200    -45.0 11.3 24    -82.5    -7.53
##  100 - 300   -117.5 11.3 24   -155.0   -80.03
##  100 - 500   -255.0 11.3 24   -292.5  -217.53
##  200 - 300    -72.5 11.3 24   -110.0   -35.03
##  200 - 500   -210.0 11.3 24   -247.5  -172.53
##  300 - 500   -137.5 11.3 24   -175.0  -100.03
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 8 estimates
```

```
plot(TukeyEmmeans$contrasts, comparisons = TRUE)
```

```
## Comparison discrepancy in group 1, 5 - 10 - 10 - 100:
##     Target overlap = 0.2667, overlap on graph = -0.0485

## Comparison discrepancy in group 1, 5 - 25 - 25 - 200:
##     Target overlap = 0.114, overlap on graph = -0.1166

## Comparison discrepancy in group 1, 5 - 100 - 10 - 25:
##     Target overlap = 0.0084, overlap on graph = -0.1876

## Comparison discrepancy in group 1, 5 - 200 - 10 - 300:
##     Target overlap = 0.0832, overlap on graph = -0.0053

## Comparison discrepancy in group 1, 5 - 300 - 300 - 500:
##     Target overlap = 0.2362, overlap on graph = -0.2824

## Comparison discrepancy in group 1, 10 - 25 - 10 - 100:
##     Target overlap = -0.1642, overlap on graph = 0.0306

## Comparison discrepancy in group 1, 10 - 300 - 100 - 500:
##     Target overlap = 0.0084, overlap on graph = -0.0048

## Comparison discrepancy in group 1, 10 - 300 - 300 - 500:
##     Target overlap = 0.3737, overlap on graph = -0.0144

## Comparison discrepancy in group 1, 25 - 50 - 50 - 200:
##     Target overlap = 0.1445, overlap on graph = -0.0621

## Comparison discrepancy in group 1, 25 - 200 - 100 - 200:
##     Target overlap = -0.1642, overlap on graph = 0.0771

## Comparison discrepancy in group 1, 50 - 100 - 100 - 300:
##     Target overlap = 0.0529, overlap on graph = -0.1784
```



6) Discuta ventajas, desventajas y alcances de cada aproximacion (predictora cuantitativa o cualitativa).

Tipo de pregunta que responde.

Cuanti. Existe una relacion lineal entre la var. resp y la var. expliactoria? Cuali. Hay una relación entre la var. resp y los niveles de la var. explicatoria?

Relación Y vs X: Cuanti. Linealidad de la respuesta Cuali. No implica linealidad de la respuesta.

Potencia: Tomando la misma variable cualitativa en vez de cuanti se pierden muchos grados de libertad, y por ende potencia en la prueba estadística.

Predicciones: Como var. cont. podés interpolar en el rango estudiado, cuando la variable es cuali no se puede

7) Pronostique por ambos modelos la concentración de cadmio absorbida por las plantas de *Thlaspi caerulescens* sometidas a 500 μM de cadmio. ¿Podría predecir la respuesta esperada a 450 μM? ¿Y si la concentración de cadmio ambiental supera los 600 μM?

• Predicciones

```
# Cd=500 con cadmio como continua y cadmio como factor.
new <- data.frame(Cd = c(500))
predict.lm(Modelo_Cdcuant, new)
```

```
##        1
## 749.4956
```

```
# Cd=450.
new <- data.frame(Cd = c(450))
predict.lm(Modelo_Cdcuant, new)
```

```
##       1
## 716.627
```

```r
# revisen: Haciendo la cuenta rapidamente para 450
coefficients(Modelo_Cdcuant)
```

```
## (Intercept)          Cd
## 420.8096928   0.6573718
```

```r
(y = 420.81 + 0.66 * 450)
```

```
## [1] 717.81
```

```r
# Haciendo la cuenta exacta para 450
coefficients(Modelo_Cdcuant)[2] * 450 + coefficients(Modelo_Cdcuant)[1]
```

```
##      Cd
## 716.627
```

¿Se puede hacer para ambos modelos? No. ¿Por qué? *Porque con Cd como factor no puedo interpolar.*

Cd=600. ¡FUERA DE RANGO!

Problema 2. Frontera agropecuaria y uso de plaguicidas

El aumento de la frontera agropecuaria y el uso de plaguicidas en la Argentina es un proceso del que aún se desconocen sus consecuencias sobre las comunidades de flora y fauna nativas. Poletta y colaboradores (2009) realizaron un trabajo cuyo objetivo fue evaluar la potencial genotoxicidad del herbicida más comúnmente utilizado, Round up (RU: glifosato), sobre eritrocitos del yacaré overo *Caiman latirostris*, luego de haber sido expuestos in ovo.Para ello embriones del yacaré fueron expuestos en estadios tempranos a concentraciones no letales del herbicida entre cero y 1750 μg/huevo. Al momento de la eclosión se tomaron muestras de sangre y se calculó el daño en el ADN mediante un índice de daño (ID). Las dosis aplicadas fueron asignadas al azar a 18 grupos de 5 huevos cada uno. Los resultados se encuentran en el archivo *Ru.txt*.

1) Indique de qué tipo de estudio se trata, la cantidad de réplicas y el tipo de variables involucradas.

Estudio experimental.
UE: Grupo de 5 huevos.
Tratamientos: 6 (concentraciones no letales de herbicida 0, 500, 750, 1000, 1250, 1750).
Replicas: 3 (se colocaron a 3 grupos con 5 huevos en cada concentración de herbicida).
VR: Indice de daño (ID). (cuantitativa, continua) (??).
VE: Concentración no letal de herbicida en μg/huevo. (cuantitativa continua (numerica)/ cualitativa (factor)).

2) Describa gráfica y estadísticamente los datos.

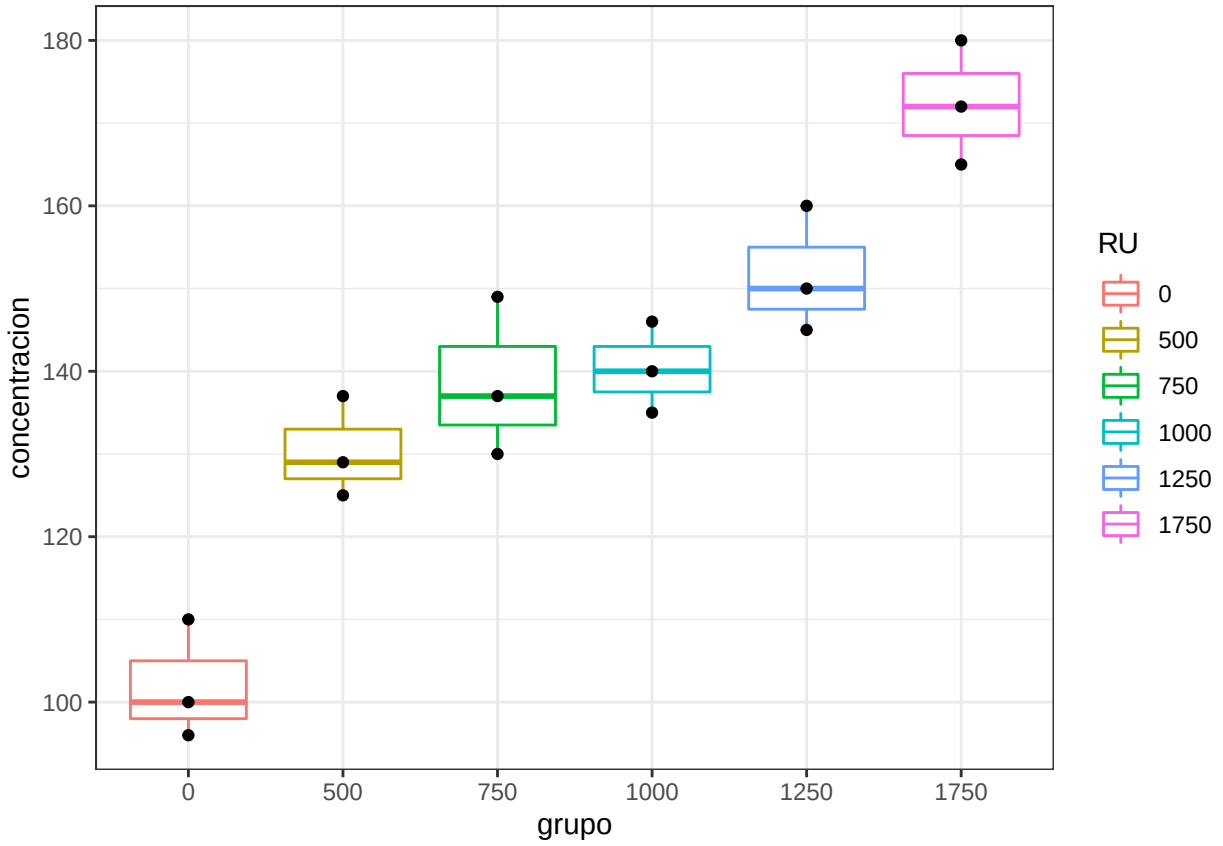
$$ID_i = \beta_0 + \beta_1 * RU[500]_{(\mu g/huevo)} + \beta_2 * RU[750]_{(\mu g/huevo)} + \beta_3 * RU[1000]_{(\mu g/huevo)} + \beta_4 * RU[1250]_{(\mu g/huevo)} + \beta_5 * RU[1750]_{(\mu g/huevo)}$$

$$+\epsilon_i$$

$$\epsilon_i \sim N(0, \sigma)$$

$$i = 1 : 18$$

```r
ru = read.table("/home/jose/Documents/materias/biome2/2019/tps/tp2/clase/Ru.txt",
    header = TRUE)
ru$RU = as.factor(ru$RU)
(box <- ggplot(ru, aes(x = RU, y = ID)) + geom_boxplot(aes(color = RU)) +
    theme_bw() + geom_point() + ylab("concentracion") + xlab("grupo"))
```



Estadística de datos separados por concentración de herbicida

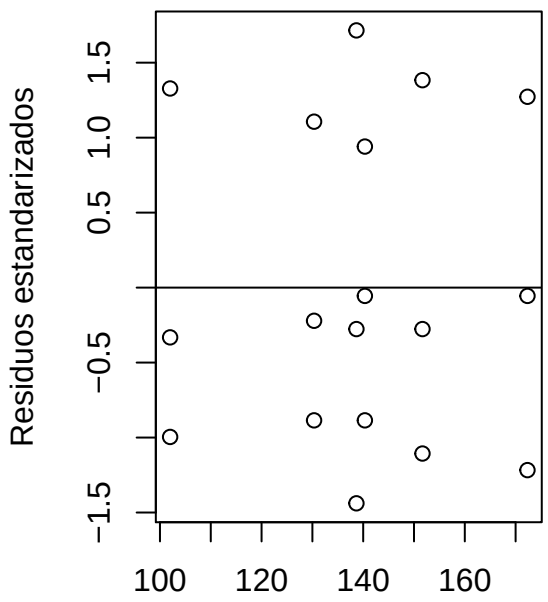
```
summarySE(ru, measurevar = "ID", groupvar = c("RU"), na.rm = TRUE)
```

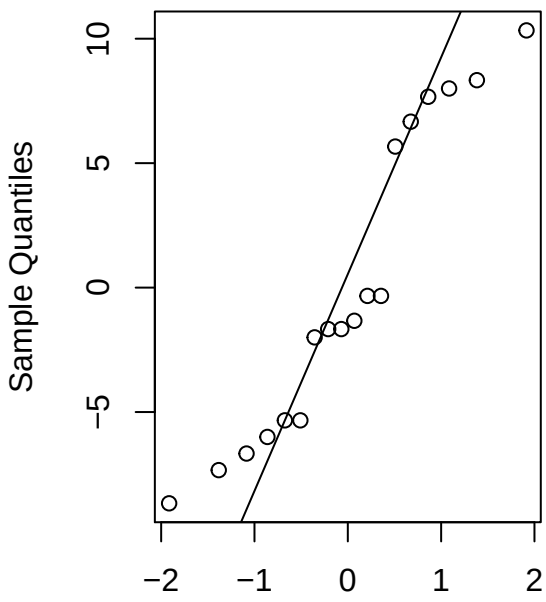
```
##     RU N       ID       sd       se       ci
## 1    0 3 102.0000 7.211103 4.163332 17.91337
## 2  500 3 130.3333 6.110101 3.527668 15.17833
## 3  750 3 138.6667 9.609024 5.547772 23.87014
## 4 1000 3 140.3333 5.507571 3.179797 13.68156
## 5 1250 3 151.6667 7.637626 4.409586 18.97292
## 6 1750 3 172.3333 7.505553 4.333333 18.64483
```

3) Analice el daño sobre el ADN en función de la concentración del herbicida. Plantee el/los modelo/s, compruebe los supuestos. Realice este procedimiento con la función *lm()*. Primero, considerando a la variable RU[μg/huevo] como *factor*. Luego, considerando a la variable RU[μg/huevo] como *numérica*.

```
mod1 = lm(ru$ID ~ ru$RU, data = ru)

e <- residuals(mod1)  # residuos
re <- rstandard(mod1)  #residuos estandarizados
pre <- predict(mod1)  #predichos
res <- data.frame(ru$RU, ru$ID, pre, e, round(re, 2))
```

```
par(mfrow = c(1, 2))
plot(pre, re, xlab = "Predichos", ylab = "Residuos estandarizados",
    main = "Gráfico de dispersión de RE vs PRED")
abline(0, 0)
qqnorm(e)
qqline(e)
```






```
par(mfrow = c(1, 1))
shapiro.test(e)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  e
## W = 0.9056, p-value = 0.072
```

```
summary(mod1)
```

```
##
## Call:
## lm(formula = ru$ID ~ ru$RU, data = ru)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.667 -5.333 -1.500  6.417 10.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  102.000      4.260  23.943 1.69e-11 ***
## ru$RU500      28.333      6.025   4.703 0.000512 ***
## ru$RU750      36.667      6.025   6.086 5.45e-05 ***
## ru$RU1000     38.333      6.025   6.363 3.60e-05 ***
## ru$RU1250     49.667      6.025   8.244 2.76e-06 ***
## ru$RU1750     70.333      6.025  11.674 6.57e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.379 on 12 degrees of freedom
## Multiple R-squared:  0.9258, Adjusted R-squared:  0.8949
## F-statistic: 29.95 on 5 and 12 DF,  p-value: 2.217e-06
```

```
anova(mod1)
```

```
## Analysis of Variance Table
##
## Response: ru$ID
##           Df Sum Sq Mean Sq F value    Pr(>F)
## ru$RU      5 8151.8 1630.36  29.945 2.217e-06 ***
## Residuals 12  653.3   54.44
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

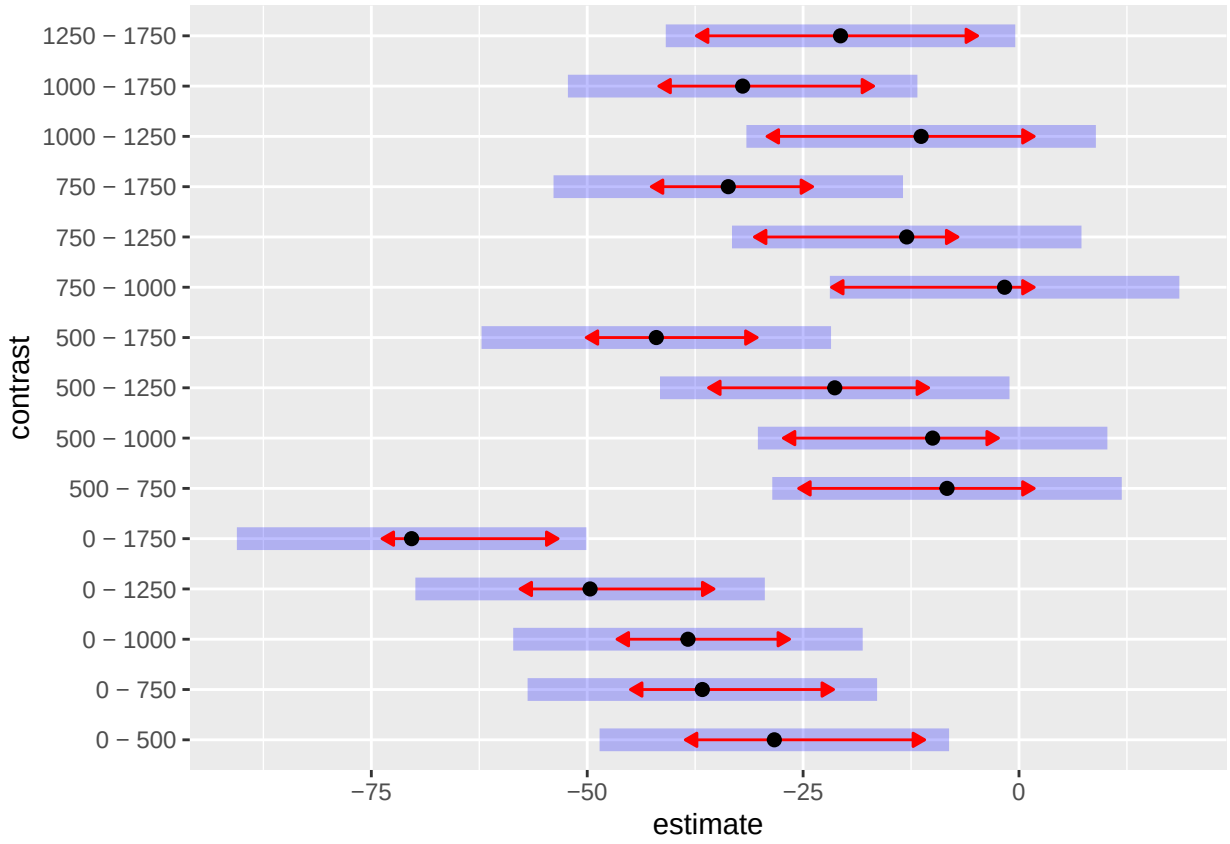
```
pval <- anova(mod1)$"Pr(>F)"[1]
```

```
library(emmeans)
(comp <- emmeans(mod1, pairwise ~ RU))
```

```
## $emmeans
##  RU   emmean   SE df t.ratio p.value
##  0       102 4.26 12 23.943  <.0001
##  500     130 4.26 12 30.594  <.0001
##  750     139 4.26 12 32.550  <.0001
##  1000    140 4.26 12 32.942  <.0001
##  1250    152 4.26 12 35.602  <.0001
##  1750    172 4.26 12 40.453  <.0001
##
##
## $contrasts
##  contrast    estimate   SE df lower.CL upper.CL
##  0 - 500       -28.33 6.02 12    -48.6    -8.10
##  0 - 750       -36.67 6.02 12    -56.9   -16.43
##  0 - 1000      -38.33 6.02 12    -58.6   -18.10
##  0 - 1250      -49.67 6.02 12    -69.9   -29.43
##  0 - 1750      -70.33 6.02 12    -90.6   -50.10
##  500 - 750      -8.33 6.02 12    -28.6    11.90
##  500 - 1000    -10.00 6.02 12    -30.2    10.24
##  500 - 1250    -21.33 6.02 12    -41.6    -1.10
##  500 - 1750    -42.00 6.02 12    -62.2   -21.76
##  750 - 1000     -1.67 6.02 12    -21.9    18.57
##  750 - 1250    -13.00 6.02 12    -33.2     7.24
##  750 - 1750    -33.67 6.02 12    -53.9   -13.43
##  1000 - 1250   -11.33 6.02 12    -31.6     8.90
##  1000 - 1750   -32.00 6.02 12    -52.2   -11.76
##  1250 - 1750   -20.67 6.02 12    -40.9    -0.43
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 6 estimates
```

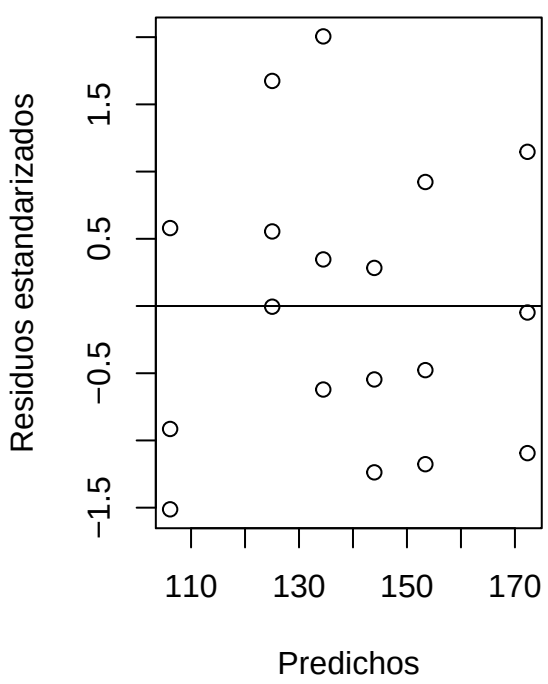
```
plot(comp$contrasts, comparisons = TRUE)
```

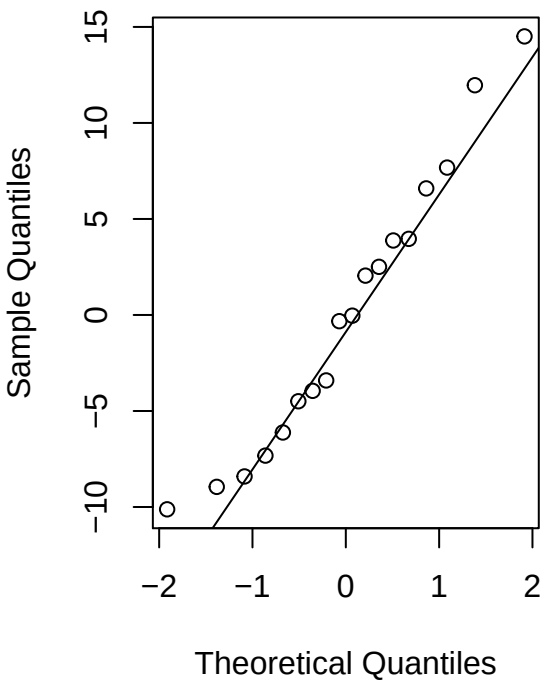
```
## Comparison discrepancy in group 1, 0 - 750 - 500 - 750:
##     Target overlap = -0.1315, overlap on graph = 0.1252

## Comparison discrepancy in group 1, 0 - 1000 - 500 - 750:
##     Target overlap = 0.1529, overlap on graph = -0.0358

## Comparison discrepancy in group 1, 0 - 1000 - 500 - 1000:
##     Target overlap = -0.1315, overlap on graph = 0.0255

## Comparison discrepancy in group 1, 0 - 1250 - 500 - 1250:
##     Target overlap = -0.1315, overlap on graph = 0.0221

## Comparison discrepancy in group 1, 500 - 1750 - 1000 - 1250:
##     Target overlap = 0.134, overlap on graph = -0.0382
```




```
# como numerica.
ru2 = read.table("/home/jose/Documents/materias/biome2/2019/tps/tp2/clase/Ru.txt",
    header = TRUE)

mod2 = lm(ru2$ID ~ ru2$RU, data = ru2)
e <- residuals(mod2)  # residuos
re <- rstandard(mod2)  #residuos estandarizados
pre <- predict(mod2)  #predichos
res <- data.frame(ru2$RU, ru2$ID, pre, e, round(re, 2))
```

```
par(mfrow = c(1, 2))
plot(pre, re, xlab = "Predichos", ylab = "Residuos estandarizados",
    main = "Gráfico de dispersión de RE vs PRED")
abline(0, 0)
qqnorm(e)
qqline(e)
```






```
par(mfrow = c(1, 1))
shapiro.test(e)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  e
## W = 0.95778, p-value = 0.5595
```

```
summary(mod2)
```

```
##
## Call:
## lm(formula = ru2$ID ~ ru2$RU, data = ru2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.1205  -5.7137  -0.1798   3.9430  14.5066
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 106.12053    3.28308   32.32 5.30e-16 ***
## ru2$RU        0.03783    0.00317   11.94 2.22e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.454 on 16 degrees of freedom
## Multiple R-squared:  0.899,  Adjusted R-squared:  0.8927
## F-statistic: 142.5 on 1 and 16 DF,  p-value: 2.224e-09
```

```
anova(mod2)
```

```
## Analysis of Variance Table
##
## Response: ru2$ID
##           Df Sum Sq Mean Sq F value    Pr(>F)
```

```
## ru2$RU     1 7916.0  7916.0  142.46 2.224e-09 ***
## Residuals 16  889.1    55.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4) Interprete cada salida en función del modelo planteado. Compare los resultados, discuta ventajas, desventajas y alcances de cada enfoque.

En el primer caso tomamos a la VE como cualitativa, es decir que estamos considerando un modelo de comparación de medias. El valor *p-valor* 2.2173455×10^{-6} indica que al menos una de las medias difiere significativamente de las demás. *b0* indica el valor estimado para el nivel de la VE que quedo de referencia (en este caso, nivel 0 μg/huevo). Despues hay 5 variables dummies, cuyo estimador indica cuánto se aleja la media del valor de la media de referencia, en unidades de la VR. Para saber cuántas medias difieren significativamente preciso realizar alguna prueba de contrastes a posteriori.

En el segundo caso tomamos a la VE como cuantitativa, realizando un modelo de regresión. La prueba para la pendiente es significativa, por lo que rechazamos que el efecto de RU sobre ID sea nulo. En este caso solo tengo *b0* y *b1*. *b0* es el valor estimado para la VR cuando la VE equivale a 0. *b1* es la pendiente que describe la relacion lineal entre la variable respuesta y variable explicatoria. Especificamente, *b1* indica cuánto cambia el índice de daño por cambio unitario en RU.

Si dejamos la VE como cuantitativa, tiene ventajas en términos de la parsimonia (menos parámetros estimados) y en la capacidad de interpolación. Lo que hacemos con el analisis es describir una relacion funcional entre las variables respuesta y explicatoria, que representa con una pendiente cuánto varia la primera en funcion del cambio unitario en la segunda. De esta manera, dentro del rango estudiado, uno puede interpolar valores predichos para la variable respuesta a valores de la VE que no hayan sido contemplados en el expermento.

Cómo utilizo a la VE depende mucho de la pregunta de investigación, Me interesa comparar esos grupos, saber si cada nivel induce una respuesta distinta en el ID? O me interesa más bien un modelo que describa cómo varía la VR en función de la VE?. Si la pregunta es relativamente ambigua, en general se elije usar la VE como cuantitativa, ya que es más informativa (permite interpolar) y más parsimoniosa (siempre dos parámetros, *b0* y *b1* más allá del número de x evaluados).

Si la relación entre VR y VE cuantitativa no es clara, o quizás si tengo evaluados pocos niveles de la VE cuantitativa podría convenir más tratar a la VE como categórica.

5) ¿Cuál es el porcentaje de variabilidad en el ID que está explicado por la concentración del herbicida?.

```r
rcuali <- round(summary(mod1)$r.squared * 100, 2)
rcuanti <- round(summary(mod2)$r.squared * 100, 2)
```

El porcentaje de variabilidad en el ID que está explicado por la concentración del herbicida para el modelo que utiliza a la VE como cualitativa es de 92.58%. Por otra parte, el porcentaje de variabilidad en el ID que está explicado por la concentración del herbicida para el modelo que utiliza a la VE como cuantitativa es de 89.9%.

6) ¿Podría predecir el ID a una concentración de RU de 1500 μg/huevo? ¿y de 2200 μg/huevo?

Podemos predecir valores de ID a niveles de la VE no evaluados en el diseño experimental sólo si la VE es cuantitativa y sólo si dicho valor se encuentra en el rango de valores puesto a prueba en el ensayo. No conocemos el compartamiento de ID fuera de ese rango, por lo que no es correcto extrapolar.

```r
# vamos a armar una funcion para predecir esos datos suponemos que
# los datos son unicos y ya estan cargados
datos = read.table("/home/jose/Documents/materias/biome2/2019/tps/tp2/clase/Ru.txt",
    header = TRUE)
prediccion <- function(valor, bd) {
    mod = lm(bd$ID ~ bd$RU, data = bd)
    b0 = mod$coefficients[1]
    b1 = mod$coefficients[2]
    res = (b1 * valor) + b0
    return(res)
}
```

```r
prediccion(valor = 1500, bd = datos)
```

```
##    bd$RU
## 162.8663
```

```r
# otra manera mucho mas sencilla
mod2
```

```
##
## Call:
## lm(formula = ru2$ID ~ ru2$RU, data = ru2)
##
## Coefficients:
## (Intercept)       ru2$RU
##   106.12053      0.03783
```

```r
# importante: la variable a ingresar debe llamarse como en el call
newVal = ref_grid(mod2, at = list(`ru2$RU` = 1500))
predict(newVal)
```

```
## [1] 162.8663
```

PARTE B: CARACTERIZACION DE LA DISTRIBUCION NORMAL

B.1 Distribucion normal y funciones de R

Comenzaremos con la distribucion normal y la utilizaremos como ejemplo para aprender las funciones de R asociadas al manejo de distribuciones de probabilidad.

Funciones

Las funciones que veremos son comunes a todas las distribuciones de probabilidad. Hay cuatro tipos de funciones, que estan determinados por la letra con la que comienzan. Seguido a esta letra se encuentra la abreviatura de la distribucion.

- Distribucion normal: 'norm'.
- Distribucion binomial: 'binom'.
- Distribucion binomial negativa: 'nbinom'.
- Distribucion de Poisson: 'pois'.

dXXX

La primera funcion que veremos es *dnorm()*. Esta funcion calcula la densidad de probabilidad de la distribucion para un valor particular de la variable, o para un vector de valores. Esto nos permite por ejemplo graficar la distribucion.

```r
# X es el punto en el que se desea calcular la densidad, mean es
# la media (parametro mu) y sd es el Desvio estandar (parametro
# sigma).
dnorm(x = 0, mean = 0, sd = 1)
```

```
## [1] 0.3989423
```

El valor obtenido NO es la probabilidad de que la variable tome valor 0. Si queremos graficar la funcion, podrias hacer lo siguiente:

- Creamos un objeto para los valores de la funcion. En caso de dudas, consultar *?seq()*.

```r
(valores_x <- seq(-4, 4, by = 0.5))
```

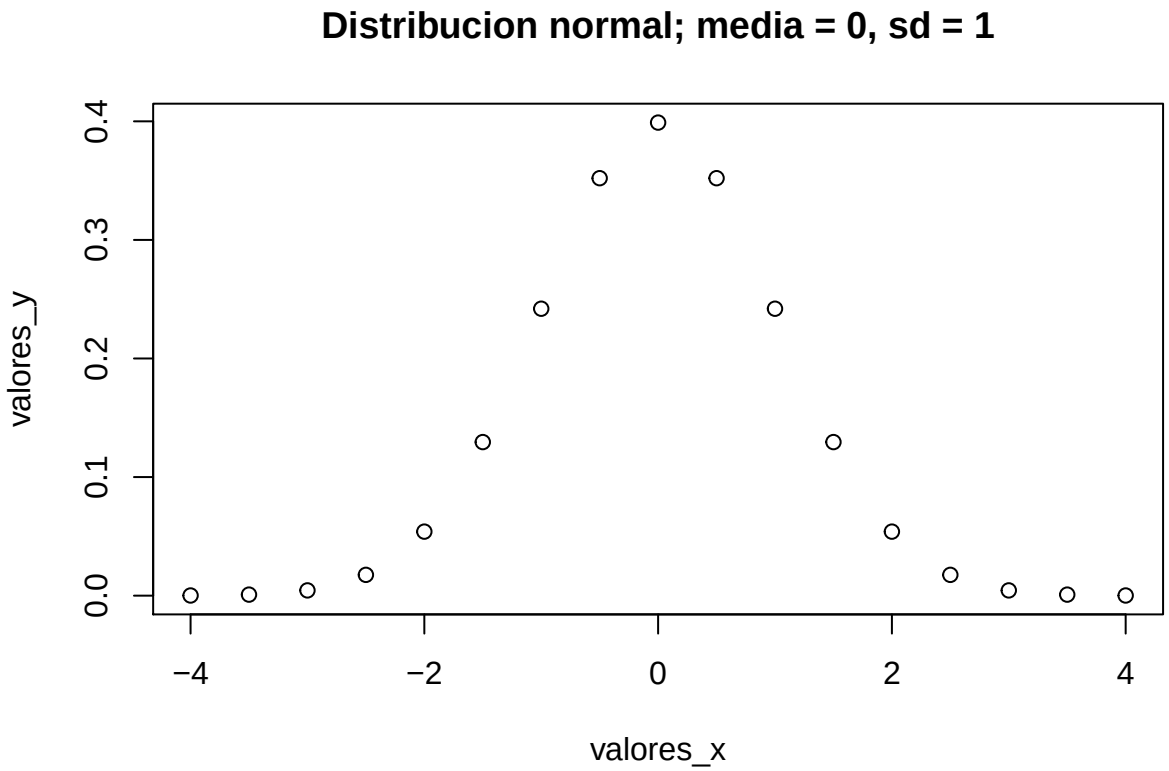
```
##  [1] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
## [16]  3.5  4.0
```

```r
(valores_y <- dnorm(x = seq(-4, 4, by = 0.5), mean = 0, sd = 1))
```

```
##  [1] 0.0001338302 0.0008726827 0.0044318484 0.0175283005 0.0539909665
##  [6] 0.1295175957 0.2419707245 0.3520653268 0.3989422804 0.3520653268
## [11] 0.2419707245 0.1295175957 0.0539909665 0.0175283005 0.0044318484
## [16] 0.0008726827 0.0001338302
```

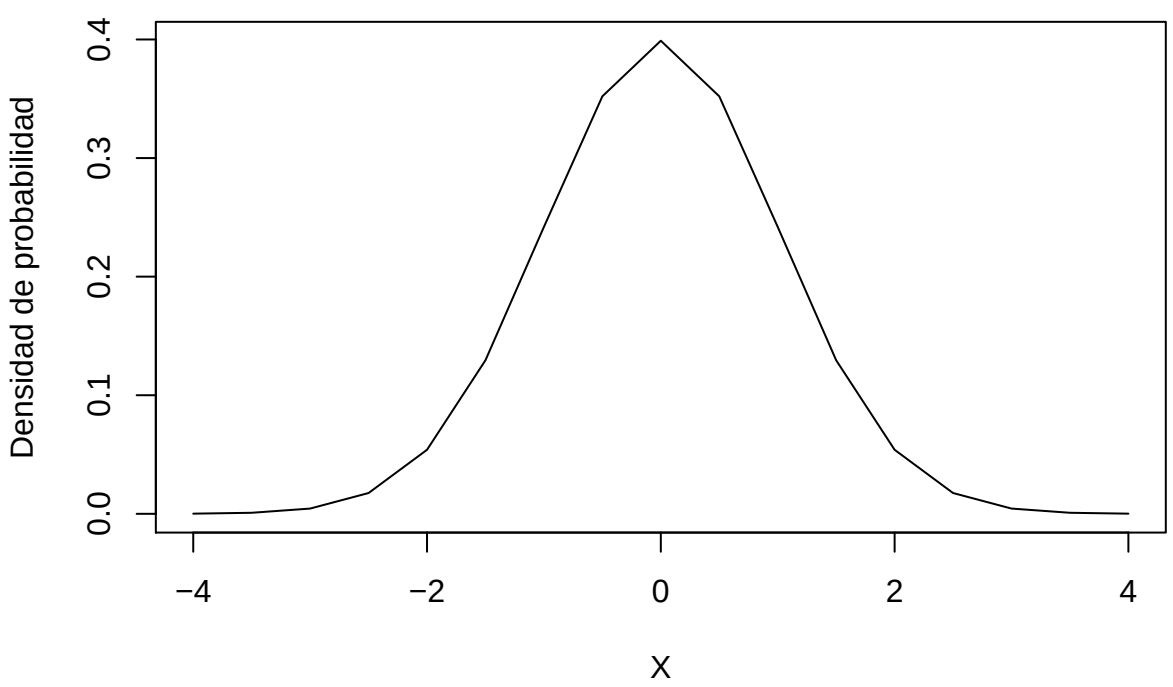
• Graficamos.

```
plot(x = valores_x, y = valores_y, main = "Distribucion normal; media = 0, sd = 1")
```



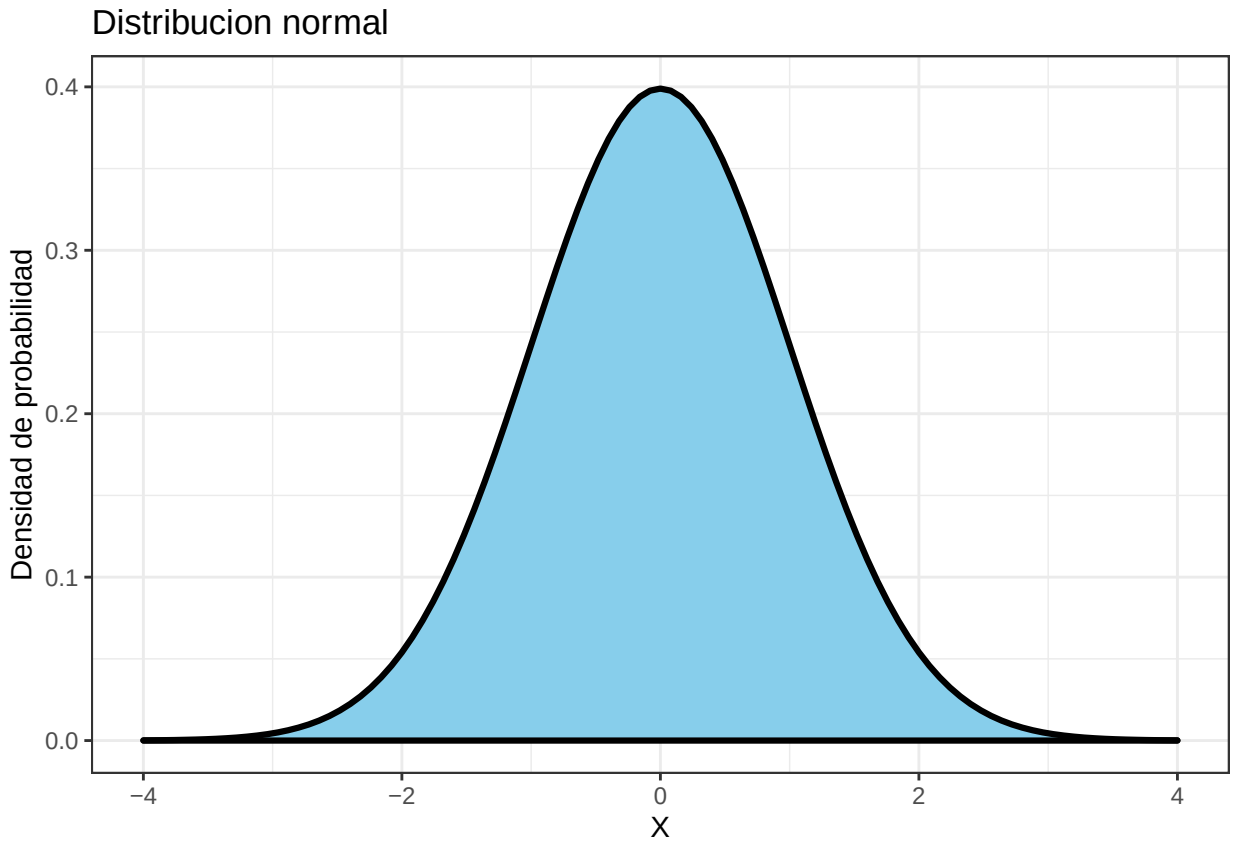
Prueben achicar el valor del argumento *by* de la funcion *seq()*.
Si especificamos que el plot sea *type = "l"*, se creara una linea uniendo los puntos que le proveamos.

```
plot(x = valores_x, y = valores_y, type = "l", main = "Distribucion normal; media = 0, sd = 1",
    xlab = "X", ylab = "Densidad de probabilidad")
```



Existen formas mas elegantes de hacer este tipo de graficos...

```
grafico <- ggplot(data.frame(x = c(-4, 4)), aes(x)) + stat_function(fun = dnorm,
    args = list(mean = 0, sd = 1), geom = "area", fill = "skyblue",
    colour = "black", lwd = 1.1) + ggtitle("Distribucion normal") +
    xlab("X") + ylab("Densidad de probabilidad") + theme_bw()
grafico
```



...pero se escapan de lo que queremos ver hoy.

pXXX

La siguente funcion que veremos es *pnorm()*. Esta funcion devuelve la probabilidad acumulada hasta el valor del argumento *q*. Al tratarse de una distribucion de probabilidad continua, esta probabilidad es la integral de la curva que se genera con *dnorm()*, entre -inf y el valor de 'q'. Si fuese una distribucion discreta, seria la sumatoria.

• Hagamos algunos calculos,

```
# Calculamos la probabilidad acumulada hasta x = 0.
pnorm(q = 0, mean = 0, sd = 1)
```

```
## [1] 0.5
```

```
# Probabilidad acumulada dentro de dos desvios estandard
# centrales.
pnorm(q = 2, mean = 0, sd = 1) - pnorm(q = -2, mean = 0, sd = 1)
```

```
## [1] 0.9544997
```

qXXX

La funcion *qnorm()* permite calcular el valor de X que acumula una dada probabilidad *p*.

• Hagamos algunos calculos,

```
# Calculamos el valor de X que acumula el 0.5 de probabilidad.
qnorm(p = 0.5, mean = 0, sd = 1)
```

```
## [1] 0
```

rXXX

La ultima funcion que vamos a ver es *rnorm()*, que genera *n* valores aleatorios, siguiendo las probabilidades establecidas por una dada distribucion normal.

• Generemos 20 valores.

```
(valoresNormales <- rnorm(n = 20, mean = 0, sd = 1))
```

```
##  [1]  0.7367062 -1.0066310 -1.7177742 -1.4292638  1.7234226 -0.4961429
##  [7] -0.1911654  0.8081055 -1.3270202 -1.1256339  1.8002943  1.9527651
## [13]  0.1560885 -0.9141164 -0.3041262  0.2661708 -0.4641820  0.6774119
## [19]  0.5144065  0.1389309
```

NOTA: los numeros aleatorios generados por el R son PSEUDOALEATORIOS. La semilla de aleatoriedad puede establecerse mediante la funcion set.seed(). Esto permite la reproduccion de los resultados.

B.2 Evaluacion de supuestos y graficos diagnosticos

En esta seccion vamos a practicar como se evaluan los supuestos de las muestras y aprenderemos a observar graficos diagnosticos.

Veamos que dice la prueba de Shapiro-Wilks sobre la muestra recien generada.

```
shapiro.test(valoresNormales)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  valoresNormales
## W = 0.95803, p-value = 0.5053
```

Ahora vamos a simular cuatro muestras de una misma distribucion normal con n de 25, 50, 100 y 200. A cada muesta le haremos un QQ-plot y realizaremos la prueba de Shapiro-Wilks.

La funcion *lapply()* toma un vector (en este caso los numeros 25, 50, 100, y 200) y les aplica una funcion, en este caso *rnorm()*. Esto significa que creara cuatro vectores, de largos 25, 50, 100, y 200, y los guardara en una lista. Los otros argumentos de la funcion *rnorm()* tambien se incluyen.

• Creamos los datos.

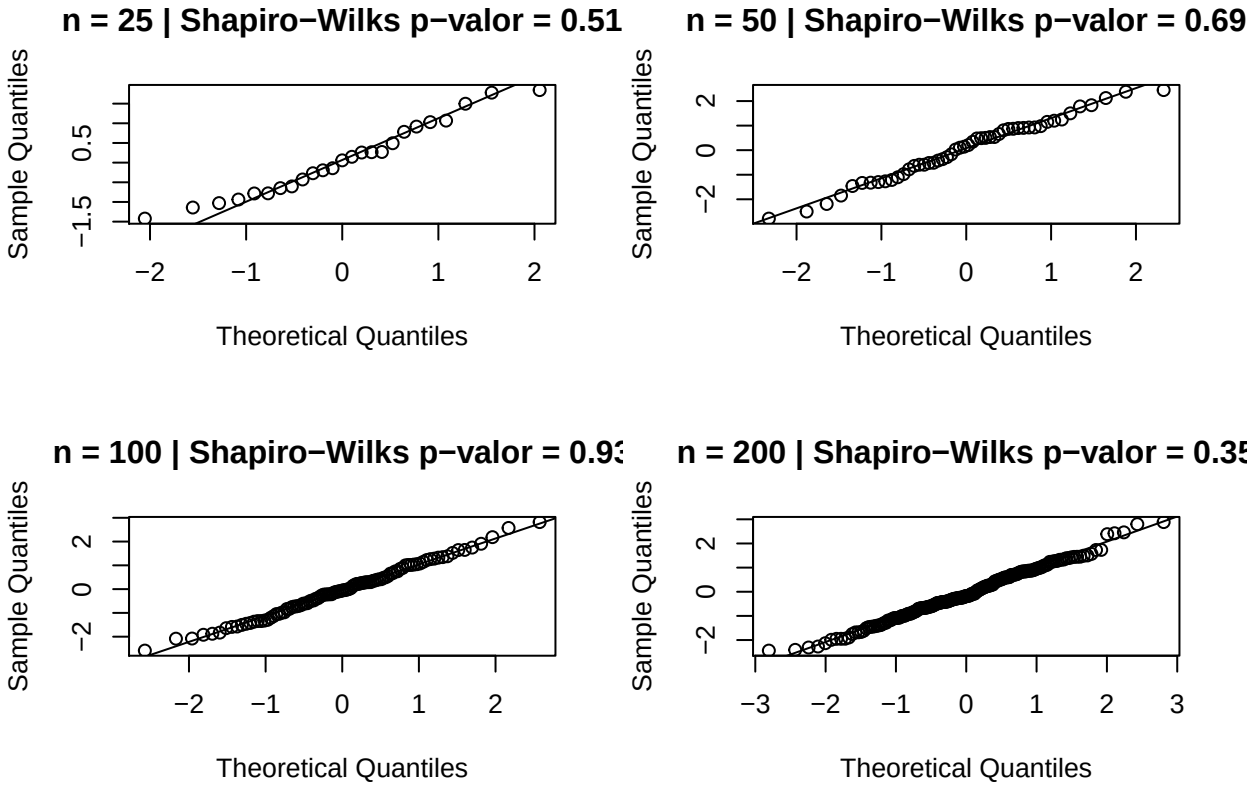
```
sampleList <- lapply(X = c(25, 50, 100, 200), FUN = rnorm, mean = 0,
    sd = 1)
```

Utilizando la misma logica, realizo la prueba de Shapiro-Wilks para cada uno de los vectores que obtuve en el paso anterior.

```
pShapiroWilks <- lapply(sampleList, shapiro.test)
```

A continuacion crearemos un QQ-plot para cada vector y mostraremos en el titulo de cada grafico el n utilizado y el p-valor de la prueba.

```
# Establecemos un arreglo grafico de dos filas y dos columnas.
par(mfrow = c(2, 2))
# Para los valores del 1 al 4 (recordar que tenemos 4 elementos en
# la lista).
for (i in 1:4) {
    qqnorm(sampleList[[i]], main = paste0("n = ", length(sampleList[[i]]),
        " | Shapiro-Wilks p-valor = ", round(pShapiroWilks[[i]]$p.value,
            digits = 2)))
    qqline(sampleList[[i]])
}
```



Si volvemos a correr la linea que crea el objeto pShapiroWilks, podremos graficar cuatro muestras diferentes.

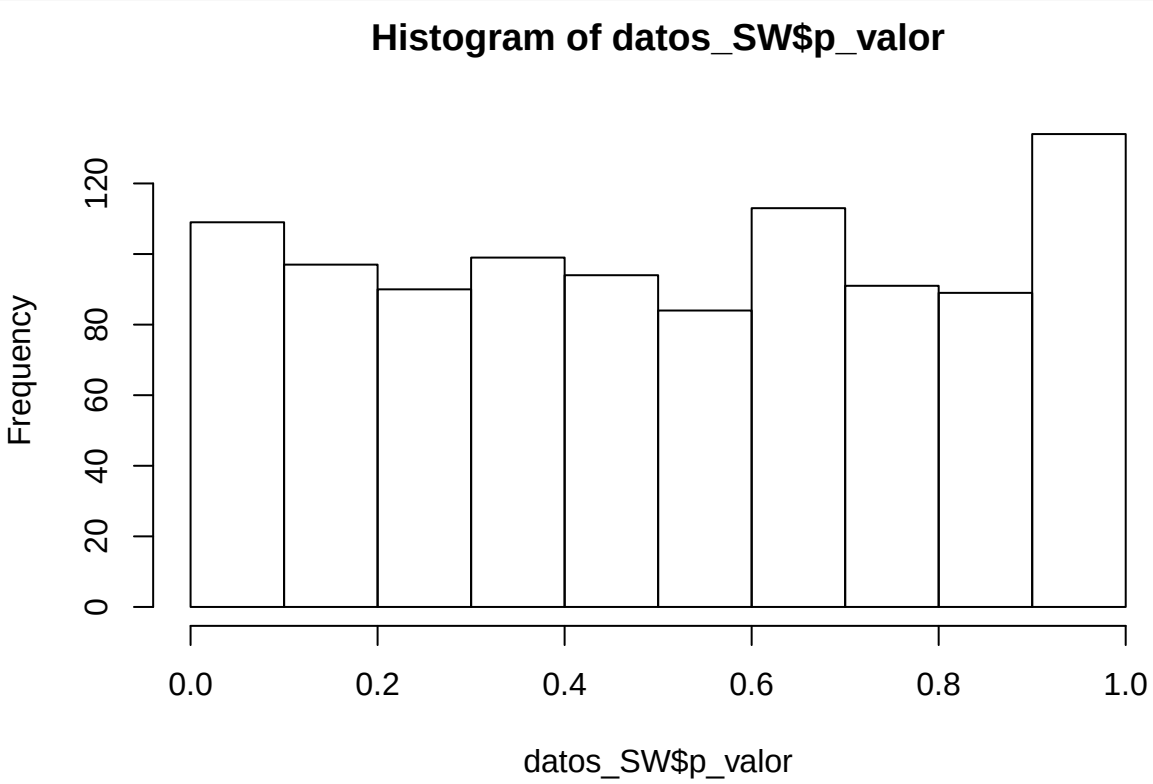
Como mencionamos en la guia, la prueba de Shapiro-Wilks es una prueba de hipotesis y utiliza por default un alfa de 0,05. Esto significa que de 100 veces, en 5 casos se rechazaria Ho.

• Vamos a simular 1000 muestras de 20 observaciones con $\mu = 30$ y $sd = 10$. Realizamos un histograma de los p-valores obtenidos y observamos en cuantos casos se rechazaria la *Ho*.

```r
# Creamos un dataframe con 2 columnas y 1000 filas.
datos_SW <- data.frame(Simulacion = vector(mode = "numeric", length = 1000),
    p_valor = vector(mode = "numeric", length = 1000))
# Hacemos un loop que extrae 1000 muestras de n = 20, ejecuta la
# prueba de Shapiro-wilks y almacena el p-valor en el dataframe.

for (i in 1:1000) {
    simulacion <- rnorm(20, 30, 10)
    p_valor_SW <- shapiro.test(simulacion)$p.value
    datos_SW[i, ] <- list(i, p_valor_SW)
}

# Veamos el dataframe.
head(datos_SW)
```

```
##   Simulacion    p_valor
## 1          1 0.70479788
## 2          2 0.65711905
## 3          3 0.09558748
## 4          4 0.30054347
## 5          5 0.34645809
## 6          6 0.85183154
```

```r
# Construyamos el histograma.
hist(datos_SW$p_valor)
```




```r
# Veamos en cuantos casos se rechaza H0.
datos_SW$conclusion <- ifelse(test = datos_SW$p_valor < 0.05, yes = "Rechazo normalidad",
    no = "No rechazo normalidad")
table(datos_SW$conclusion)
```

```
##
## No rechazo normalidad    Rechazo normalidad
##                   945                    55
```

En la siguiente seccion graficaremos los QQ-plots de muestras provenientes de una poblacion normal que no superan la

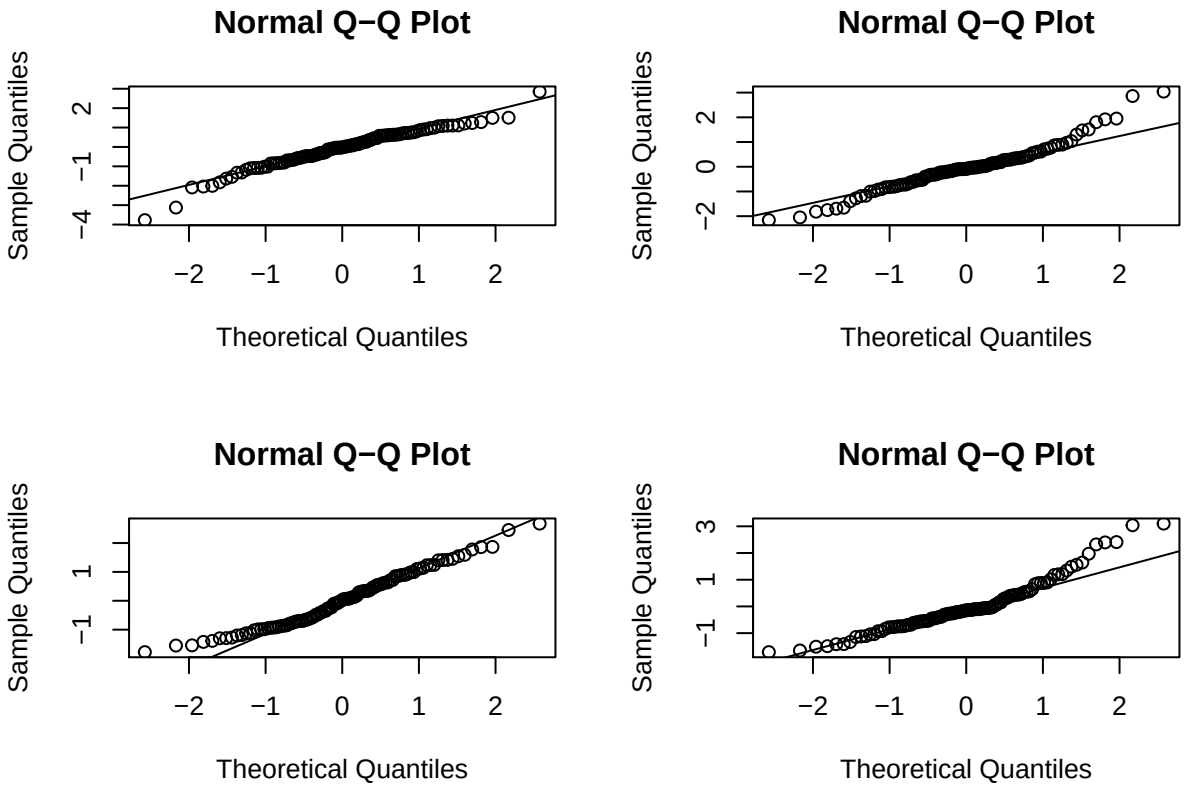
prueba de Shapiro-Wilks.

• Generamos muestras de una distribucion normal con p-valores de Shapiro-Wilks $< 0,05$.

```
# Simulemos.
muestras <- vector(mode = "list", length = 4)
# Para cada elemento de 'muestras, creo un vector con la muestra y
# obtengo el p-valor.  Mientras el p-valor sea >= 0.05 creo un
# nuevo vector, obtengo el p-valor y vuelvo a evaluar la
# condicion.  Cuando la condicion deja de cumplirse guardo la
# muestra x en el objeto 'muestra'
for (i in seq_along(muestras)) {
    x <- rnorm(n = 100, mean = 0, sd = 1)
    p_valor <- shapiro.test(x)$p.value
    while (p_valor >= 0.05) {
        x <- rnorm(n = 100, mean = 0, sd = 1)
        p_valor <- shapiro.test(x)$p.value
    }
    muestras[[i]] <- x
}
```

• Ahora grafiquemos los QQ-plots de las cuatros muestras.

```
par(mfrow = c(2, 2))
for (i in seq_along(muestras)) {
    qqnorm(muestras[[i]])
    qqline(muestras[[i]])
}
```



Si repiten ambos loops pueden observar diferentes muestras de a cuatro para ajustar un poco el analisis de graficos.

```
par(mfrow = c(1, 1))  # Restauramos la cantidad de graficos por dispositivo.
```

A.3 Simulaciones en regresion

Ahora queremos ver que es lo que pasa con las estimaciones cuando simulamos regresiones normales. Para esto primero vamos a ver como generar artificialmente una regresion lineal simple.

Una regresion tiene la siguiente forma:

$$Y = \beta_0 + \beta_1 * x + \epsilon$$

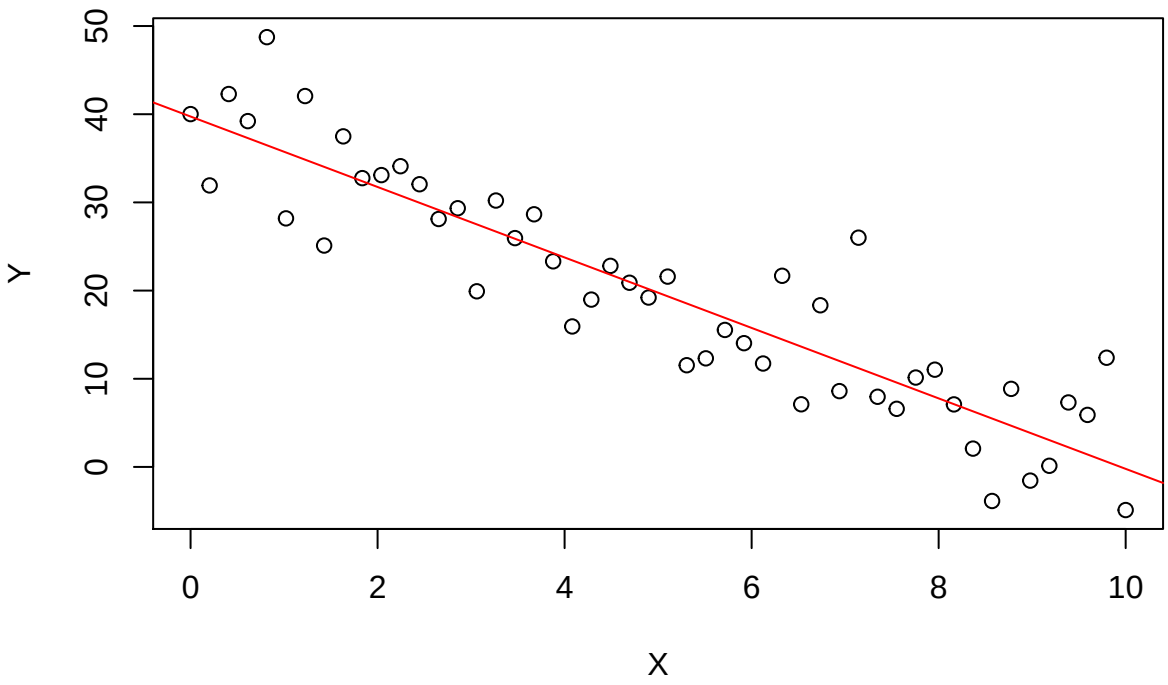
• Simule una regresion lineal simple con tamaño de la muestra de 50, β_0=40, β_1=-4 y σ=5. Grafique.

Sugerencia: para generar los ϵ utilice la funcion rnorm. Recuerde que la funcion lm ajusta un modelo lineal

```
beta0 = 40
beta1 = -4
sigma = 5
n = 50
X <- seq(0, 10, length = n)
e = rnorm(n, 0, sigma)
Y = beta0 + beta1 * X + e
m1 <- lm(Y ~ X)
```

• Grafique para inspeccionar los datos.

```
plot(X, Y)
abline(lm(Y ~ X), col = "red")
```



• Inspeccione el resultado del modelo utilizando la funcion *summary()*

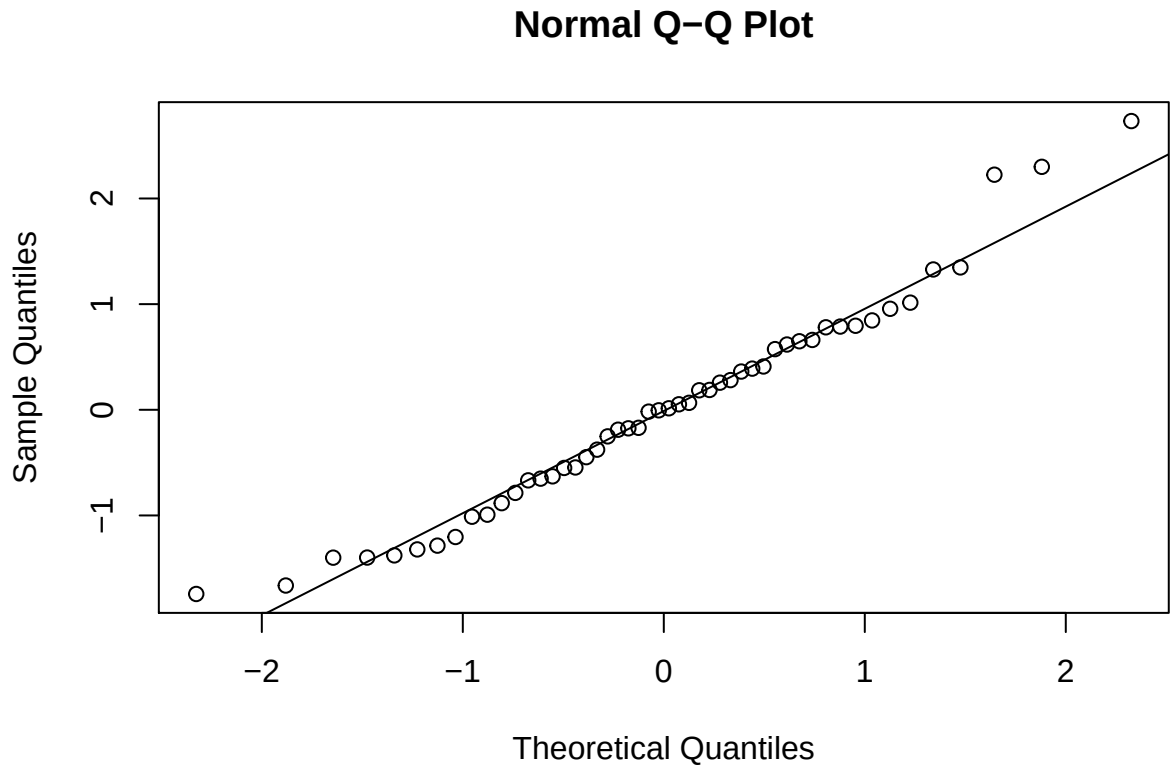
```
summary(m1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3501 -3.6122  0.0245  3.4740 14.8012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

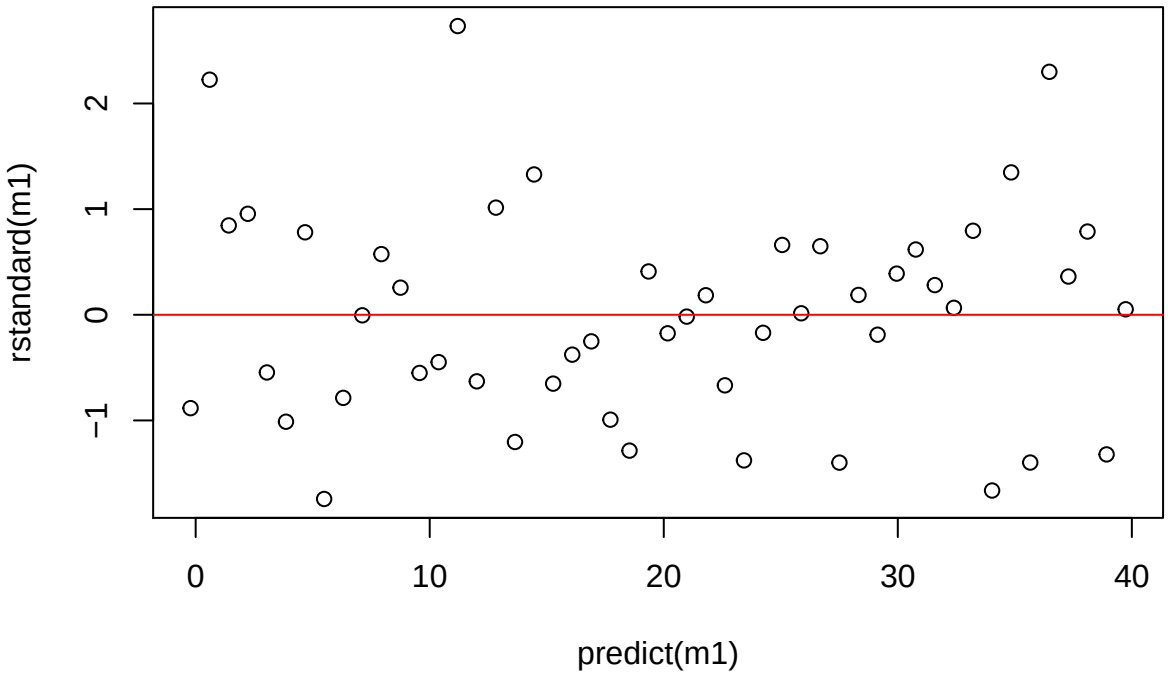
```
## (Intercept)  39.7363     1.5330   25.92   <2e-16 ***
## X            -3.9955     0.2642  -15.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.501 on 48 degrees of freedom
## Multiple R-squared:  0.8266, Adjusted R-squared:  0.8229
## F-statistic: 228.7 on 1 and 48 DF,  p-value: < 2.2e-16
```

• Verifique los supuestos de normalidad y homogeneidad de varianzas. Ayuda: utilice la funcion *rstandard* y *predict*.

```r
# normalidad
em1 <- rstandard(m1)
qqnorm(em1)
qqline(em1)
```




```r
# homogeneidad de varianzas
plot(predict(m1), rstandard(m1))
abline(0, 0, col = "red")
```

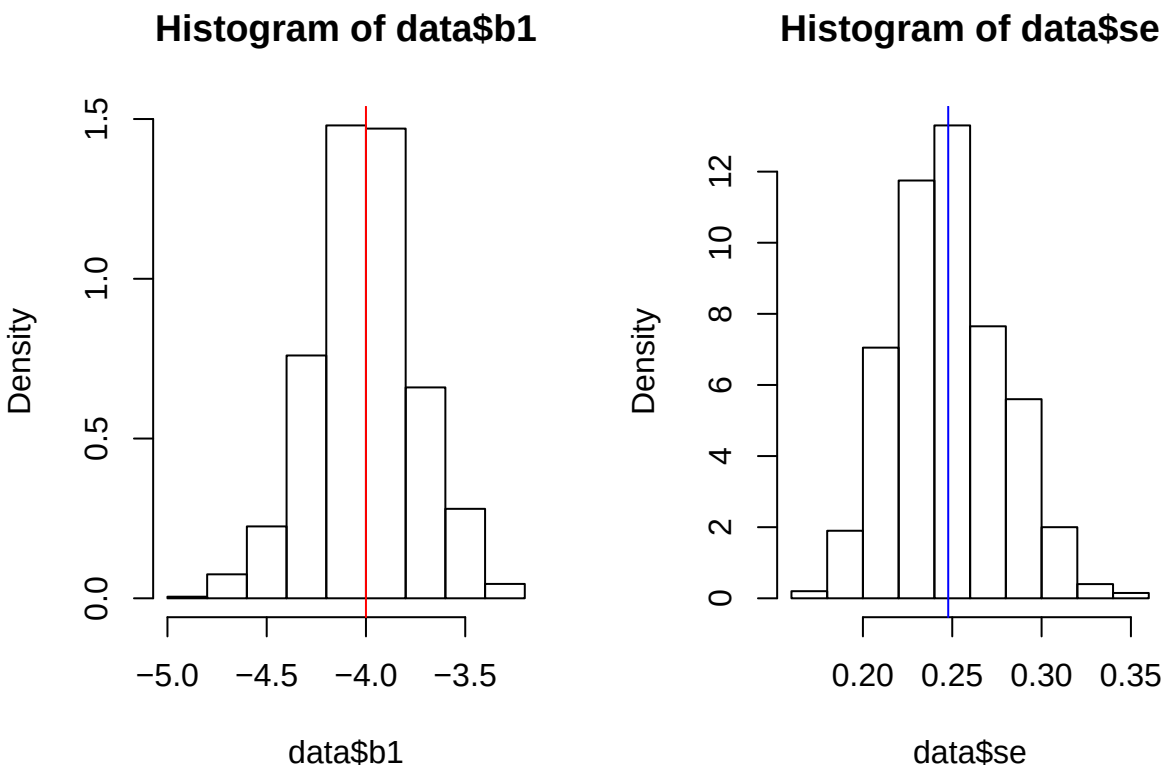


• Repita la simulacion del modelado 1000 veces, extraiga el coeficiente β_1 en cada simulacion y realice un histograma. Ayuda: para obtener el valor del coeficiente haga *model$coefficients[2].*

Mas ayuda: aca debajo les dejamos una forma de simular varias veces algun proceso que uno desea y lo guarda* en un data frame.**

```
data <- data.frame()
for (i in 1:1000) {
    X <- runif(n, 0, 10)
    e = rnorm(n, 0, sigma)
    Y = beta0 + beta1 * X + e
    model <- lm(Y ~ X)
    b1 <- c(i, model$coefficients[2], summary(model)[[4]][4])
    data <- cbind(rbind(data, b1))
}
colnames(data) <- c("simulacion", "b1", "se")
```

```
par(mfrow = c(1, 2))
hist(data$b1, freq = FALSE)
abline(v = beta1, col = "red")
hist(data$se, freq = FALSE)
abline(v = mean(data$se), col = "blue")
```

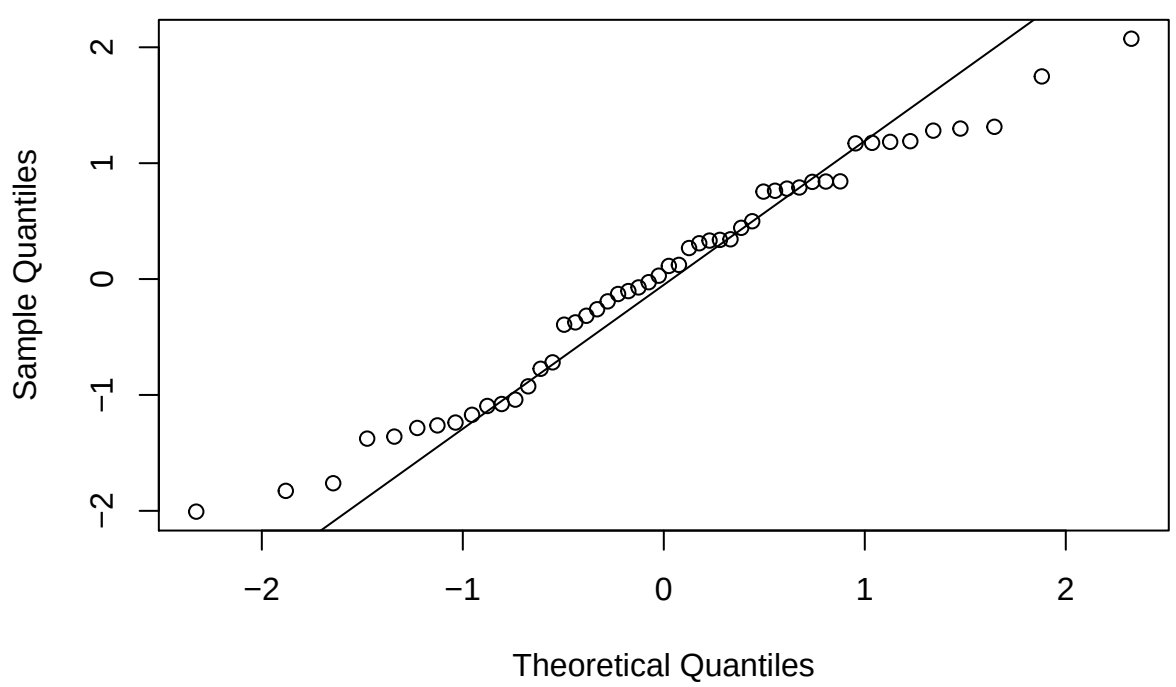


• ¿Qué ocurre si aumentamos el error del modelo (varianza residual, no explicada)?. Realice el mismo procedimiento que antes pero con σ=50 y justifique.

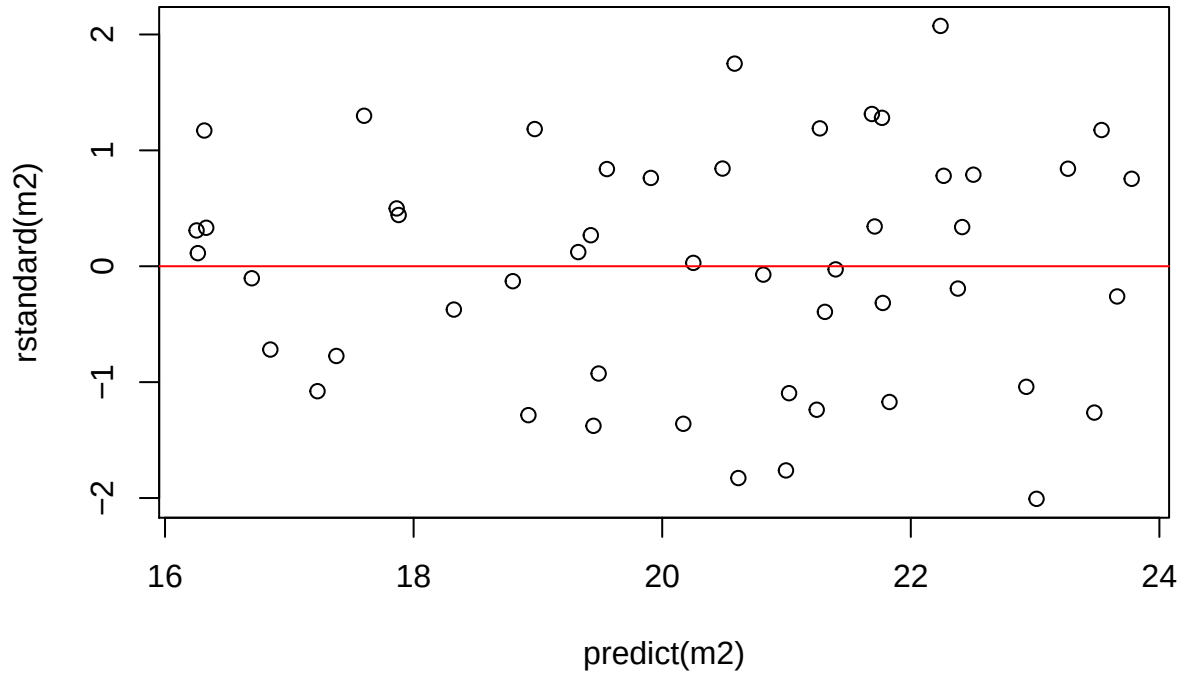
```
beta0 = 40
beta1 = -4
sigma2 = 50
n = 50
X2 <- seq(0, 10, length = n)
e2 = rnorm(n, 0, sigma2)
Y = beta0 + beta1 * X2 + e
m2 <- lm(Y ~ X)
summary(m2)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.5770 -10.5480   0.8258   9.3083  24.5542
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.9375     3.7493   4.251 9.76e-05 ***
## X             0.7870     0.6074   1.296    0.201
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.06 on 48 degrees of freedom
## Multiple R-squared:  0.0338, Adjusted R-squared:  0.01367
## F-statistic: 1.679 on 1 and 48 DF,  p-value: 0.2013
```

```
# normalidad
em2 <- rstandard(m2)
qqnorm(em2)
qqline(em2)
```

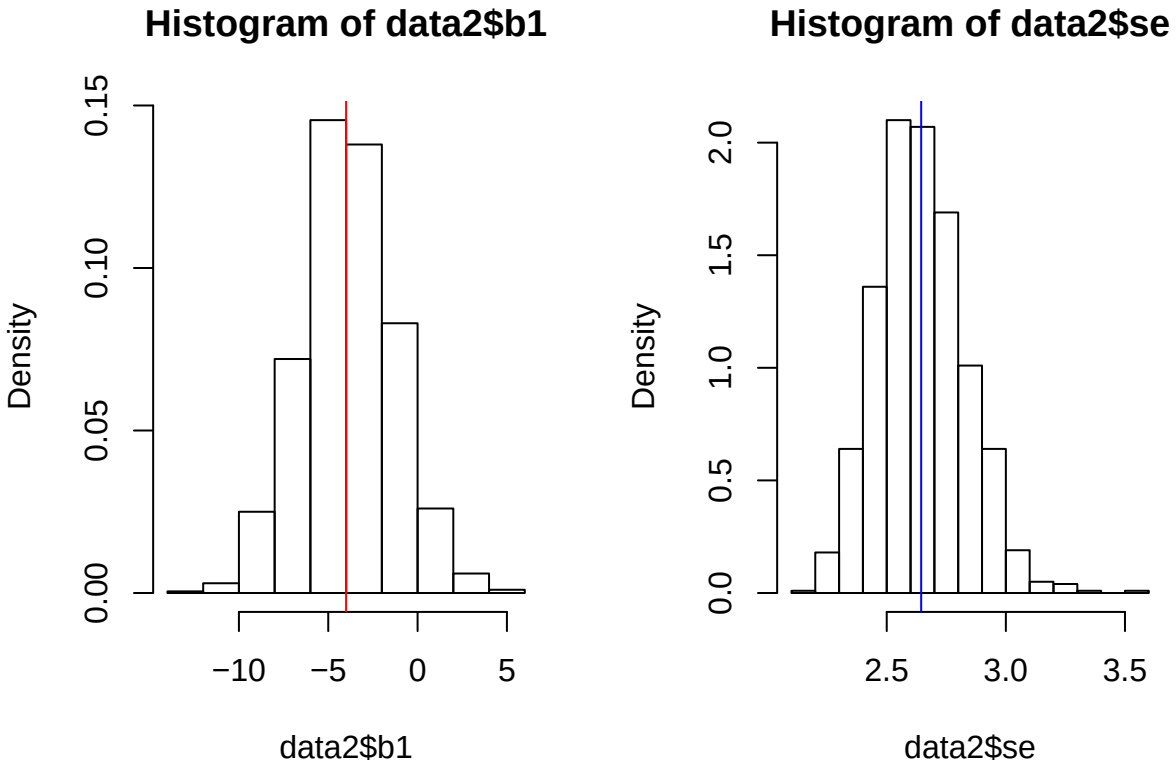



```r
# homogeneidad de varianzas
plot(predict(m2), rstandard(m2))
abline(0, 0, col = "red")
```




```r
data2 <- data.frame()
for (i in 1:1000) {
    X <- runif(n, 0, 10)
    e = rnorm(n, 0, sigma)
    Y = beta0 + beta1 * X + e2
    model <- lm(Y ~ X)
    b2 <- c(i, model$coefficients[2], summary(model)[[4]][4])
    data2 <- cbind(rbind(data2, b2))
}
```

```r
colnames(data2) <- c("simulacion", "b1", "se")
par(mfrow = c(1, 2))
hist(data2$b1, freq = FALSE)
abline(v = beta1, col = "red")
hist(data2$se, freq = FALSE)
abline(v = mean(data2$se), col = "blue")
```



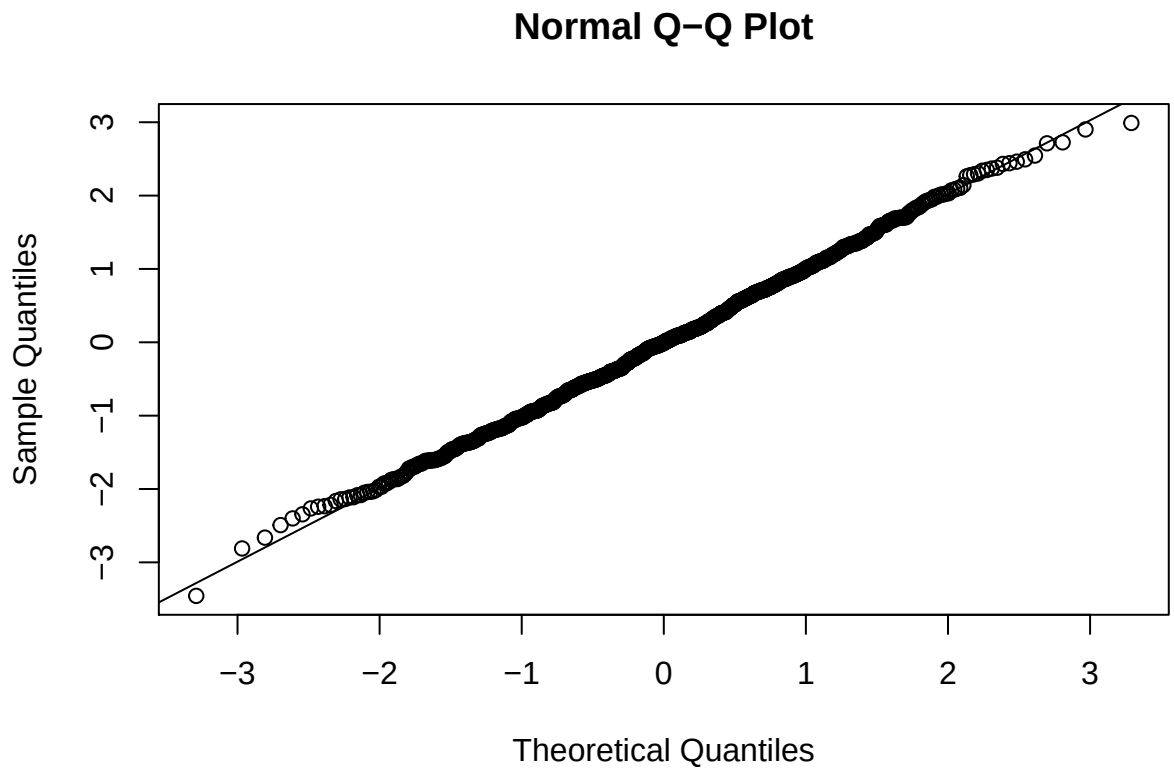
• ¿Qué ocurre si aumentamos el n?. Realice el mismo procedimiento que antes pero con n=1000 y justifique.

```r
beta0 = 40
beta1 = -4
sigma = 5
n2 = 1000
X <- seq(0, 10, length = n2)
e3 = rnorm(n2, 0, sigma)
Y = beta0 + beta1 * X + e3
m3 <- lm(Y ~ X)
summary(m3)
```

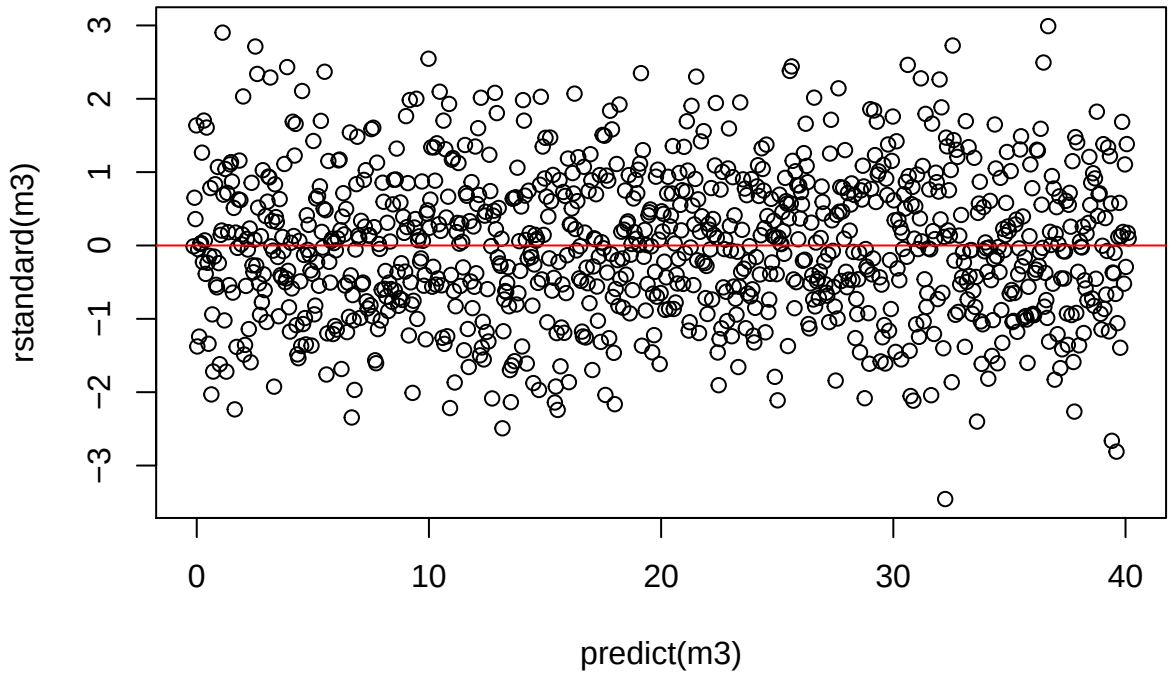
```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.561  -3.345  -0.046   3.526  15.183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 40.13613    0.32138  124.89   <2e-16 ***
## X           -4.02734    0.05565  -72.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.085 on 998 degrees of freedom
```

```
## Multiple R-squared:  0.8399, Adjusted R-squared:  0.8398
## F-statistic:  5237 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
# normalidad
em3 <- rstandard(m3)
qqnorm(em3)
qqline(em3)
```

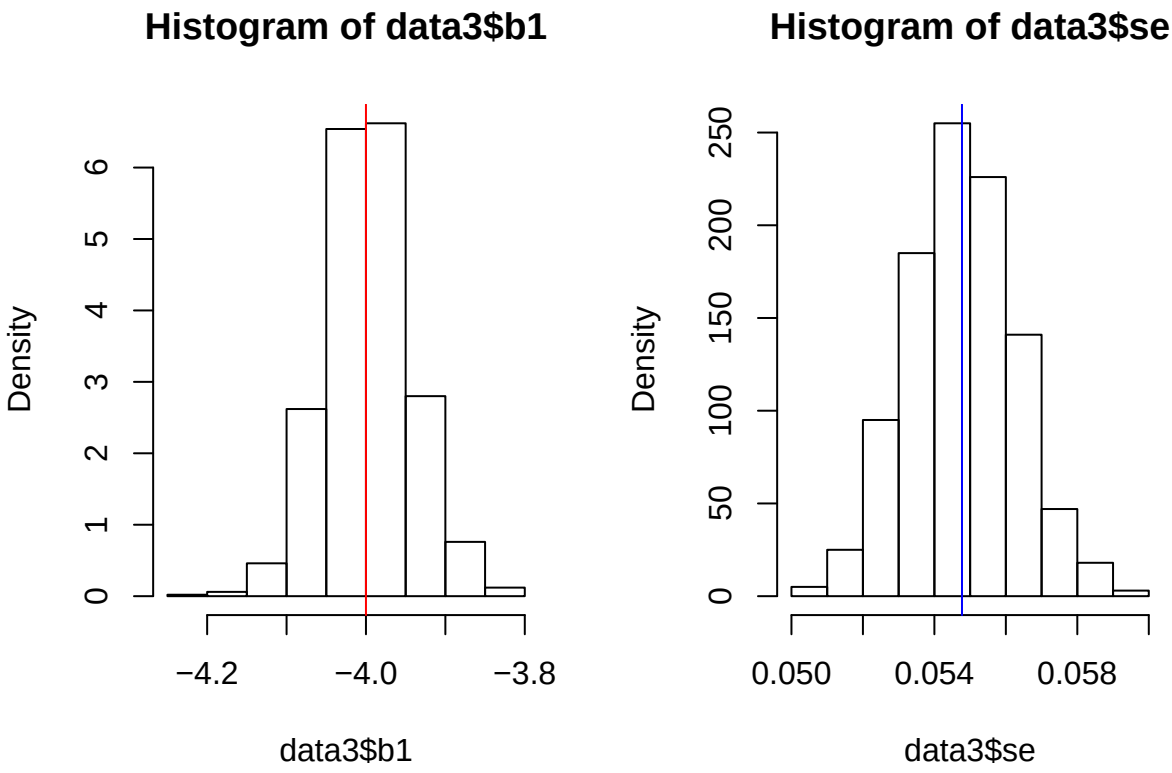



```r
# homogeneidad de varianzas
plot(predict(m3), rstandard(m3))
abline(0, 0, col = "red")
```




```r
data3 <- data.frame()
for (i in 1:1000) {
```

```
    X <- runif(n2, 0, 10)
    e = rnorm(n2, 0, sigma)
    Y = beta0 + beta1 * X + e
    model <- lm(Y ~ X)
    b3 <- c(i, model$coefficients[2], summary(model)[[4]][4])
    data3 <- cbind(rbind(data3, b3))
}
colnames(data3) <- c("simulacion", "b1", "se")
par(mfrow = c(1, 2))
hist(data3$b1, freq = FALSE)
abline(v = beta1, col = "red")
hist(data3$se, freq = FALSE)
abline(v = mean(data3$se), col = "blue")
```



• Hacemos un resumen.

```
res <- data.frame(cbind(rep(beta0, 3), rep(beta1, 3), c(sigma, sigma2,
    sigma), c(n, n, n2), c(mean(data$b1), mean(data2$b1), mean(data3$b1)),
    c(mean(data$se), mean(data2$se), mean(data3$se))))
colnames(res) <- c("b0", "b1", "sigma", "n", "mean(b1)", "mean(se)")
res
```

```
##   b0 b1 sigma    n  mean(b1)   mean(se)
## 1 40 -4     5   50 -4.006665 0.24774213
## 2 40 -4    50   50 -3.908716 2.64456026
## 3 40 -4     5 1000 -3.996840 0.05477229
```

Ahora bien, ¿Que pasa si el supuesto de homogeneidad de varianzas no se cumple? ¿Como se visualiza?

• Simule la misma regresion pero ahora calcule a ϵ como: $\epsilon = rnorm(n, 0, sigma * X)$. Verifique los supuestos y grafique.

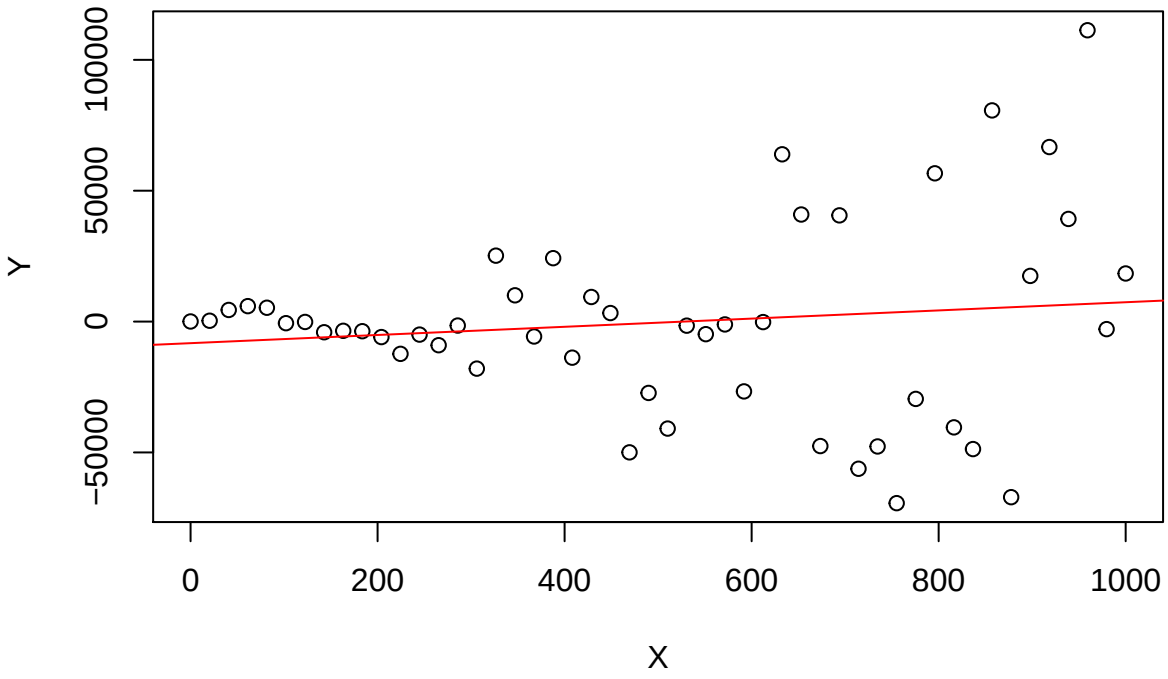
```
beta0 = 40
beta1 = -4
sigma = 50
n = 50
X <- seq(0, 1000, length = n)
e = c()
```

```r
for (i in 1:n) {
    e = c(e, rnorm(1, 0, sigma * X[i]))
}
Y = beta0 + beta1 * X + e
m1 <- lm(Y ~ X)
summary(m1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -72937 -13865   1816  12207 104553
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8253.86   10255.24  -0.805    0.425
## X              15.64      17.67   0.885    0.380
##
## Residual standard error: 36800 on 48 degrees of freedom
## Multiple R-squared:  0.01606,    Adjusted R-squared:  -0.004435
## F-statistic: 0.7837 on 1 and 48 DF,  p-value: 0.3804
```

• Graficamos.

```r
plot(X, Y)
abline(lm(Y ~ X), col = "red")
```



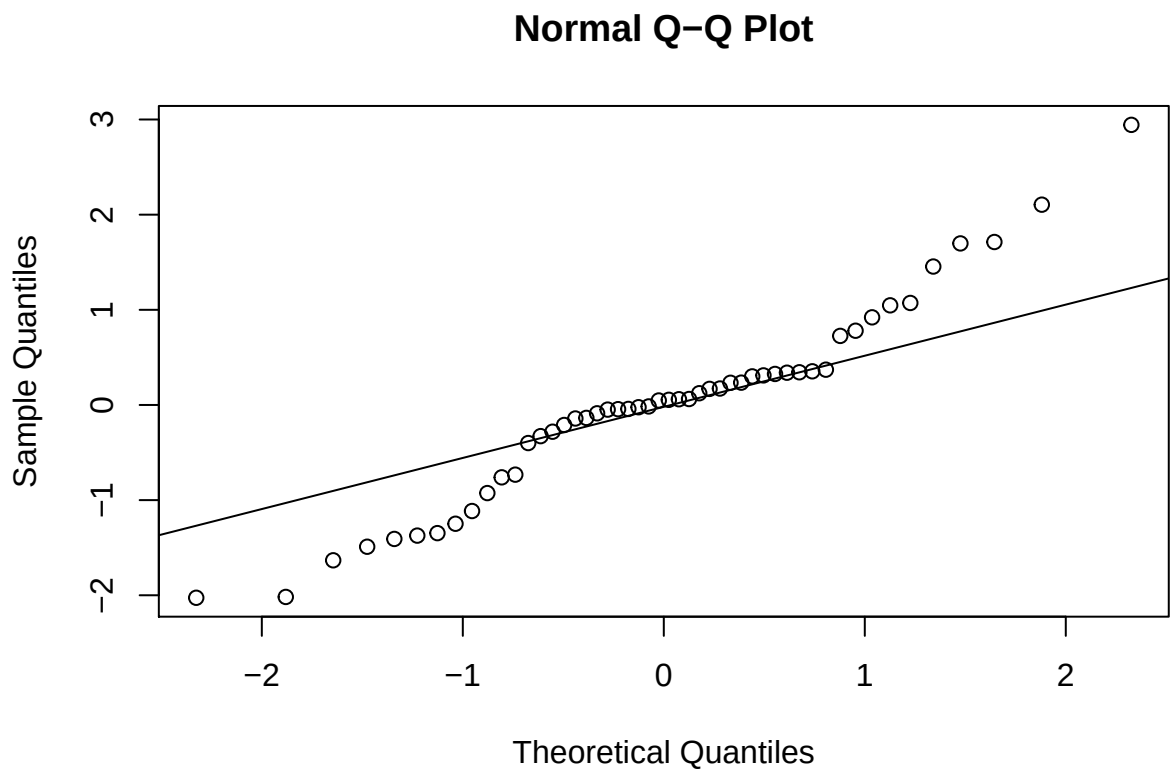
• Verificamos supuestos.

```r
# normalidad
em1 <- rstandard(m1)
shapiro.test(em1)
```

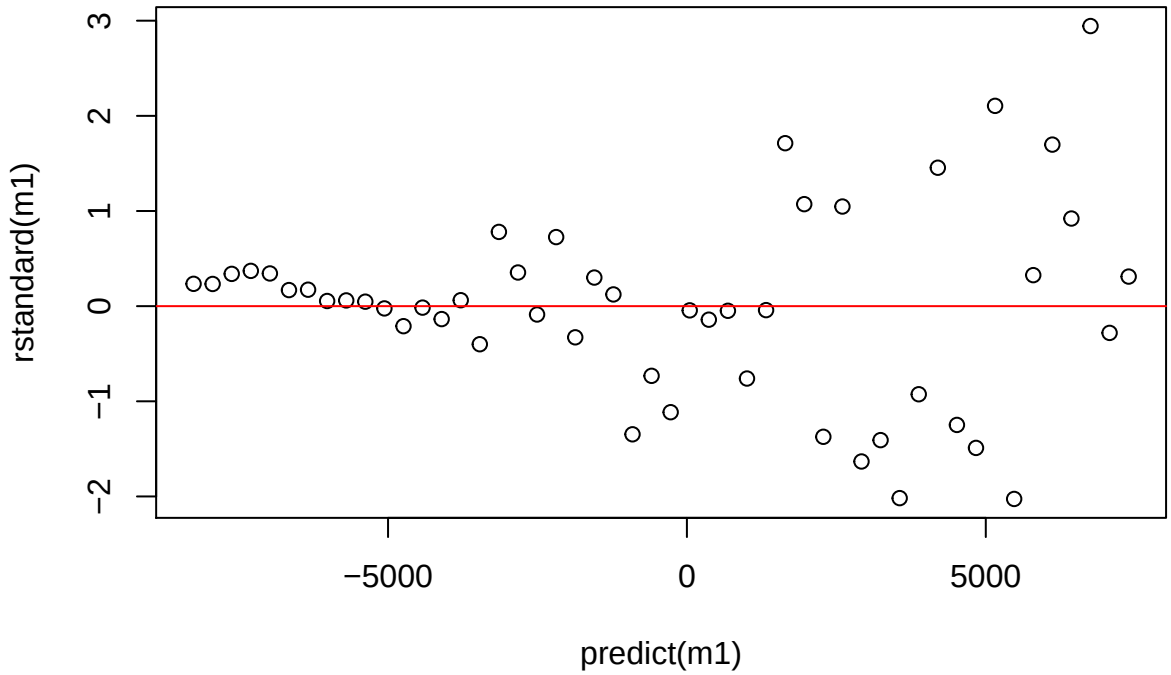
```
##
##  Shapiro-Wilk normality test
##
## data:  em1
```

```
## W = 0.95779, p-value = 0.07184
```

```r
qqnorm(em1)
qqline(em1)
```




```r
# homogeneidad de varianzas
plot(predict(m1), rstandard(m1))
abline(0, 0, col = "red")
```



Vemos que no se cumple ni la normalidad ni la homogeneidad de varianzas, y eso se debe a que la varianza de ϵ es funcion lineal de la variable predictora.

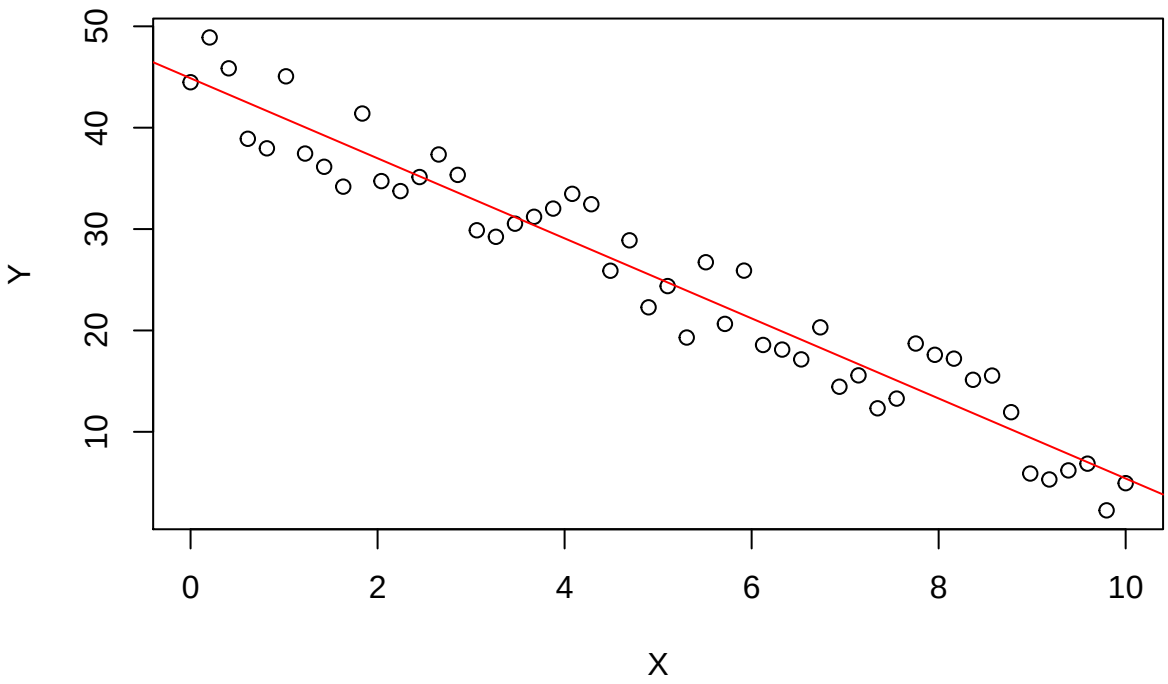
• Simule la misma regresion pero ahora calcule a ϵ como: $\epsilon = runif(n, 0, 10)$. Verifique los supuestos y grafique.

```r
beta0 = 40
beta1 = -4
sigma = 5
n = 50
X <- seq(0, 10, length = n)
e = runif(n, 0, 10)
Y = beta0 + beta1 * X + e
m1 <- lm(Y ~ X)
summary(m1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6256 -2.7094 -0.5542  2.8638  4.8296
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  44.8778     0.8633   51.99   <2e-16 ***
## X            -3.9483     0.1488  -26.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.098 on 48 degrees of freedom
## Multiple R-squared:  0.9362, Adjusted R-squared:  0.9349
## F-statistic: 704.4 on 1 and 48 DF,  p-value: < 2.2e-16
```

Graficamos

```r
plot(X, Y)
abline(lm(Y ~ X), col = "red")
```

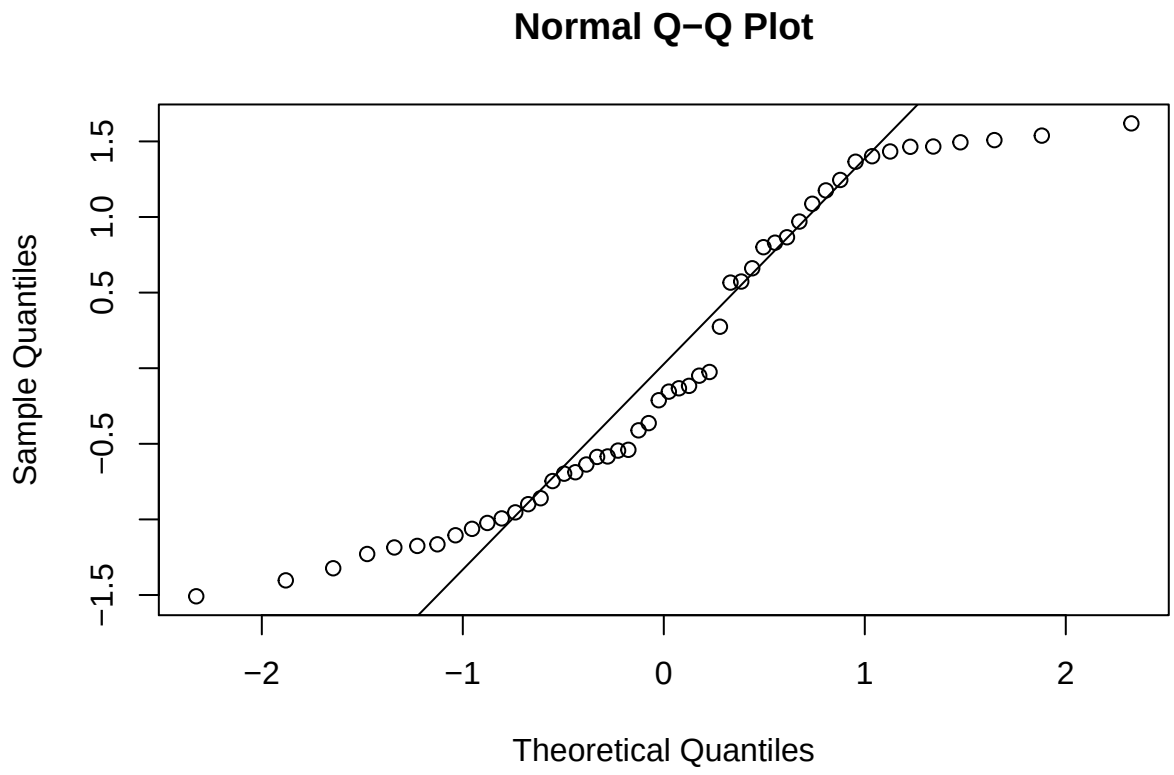


Verificamos supuestos

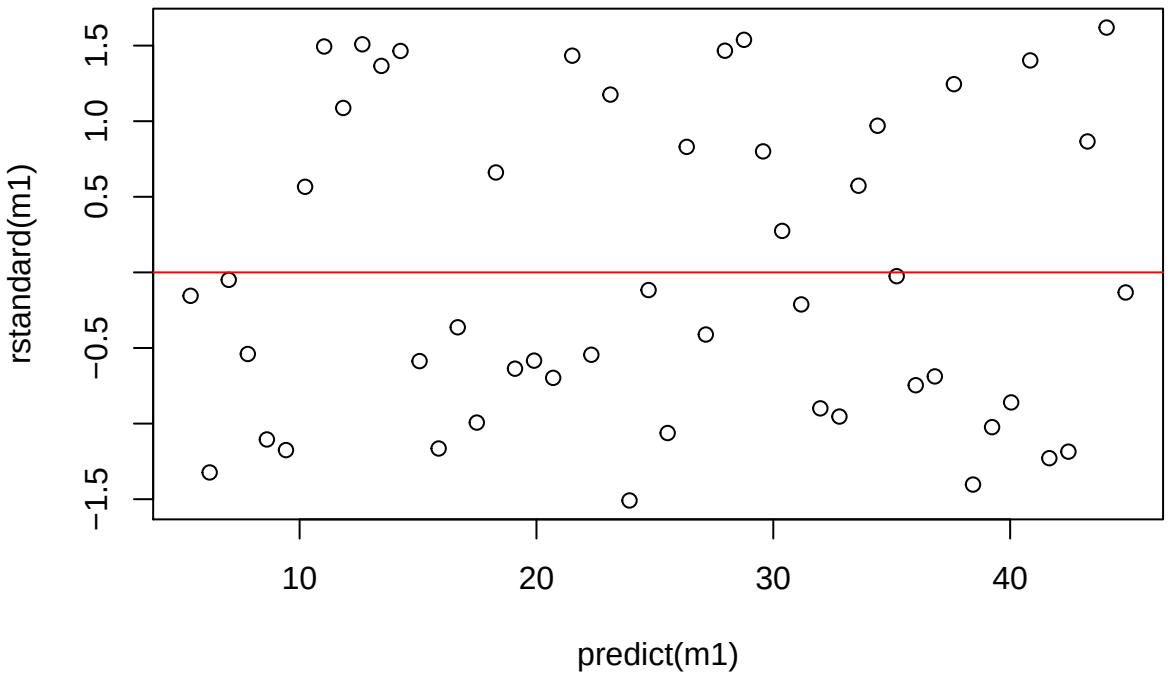
```r
# normalidad
em1 <- rstandard(m1)
shapiro.test(em1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  em1
## W = 0.90576, p-value = 0.0007488
```

```r
qqnorm(em1)
qqline(em1)
```




```r
# homogeneidad de varianzas
plot(predict(m1), rstandard(m1))
abline(0, 0, col = "red")
```



Vemos que no se cumple normalidad pero si la homogeneidad de varianzas, y eso se debe a que ϵ no proviene de una

distribucion normal.