

# Metodología Cuantitativa Avanzada I

## Fundamentos de los modelos lineales

Carrasco, D., PhD

Centro de Medición MIDE UC

PSI4035

Santiago, Marzo 16 de 2022

Taller

# Modelos de regresión

Ajuste de modelos de regresión bivariada

# Taller 1: Fundamentos de los de los modelos lineales

En esta sesión vamos a ilustrar empleando a R diferentes conceptos y operaciones introducidos por Vik (2014) en los capítulos 1 a 3.

Los conceptos que vamos a revisar son:

- variable dependiente o variable de respuesta
- variable independiente, covariante o variable predictora
- ecuación de un modelo de regresión
- suma de errores al cuadrado
- evaluar supuestos de un modelo de regresión
  - linealidad
  - homocedasticidad
  - normalidad de los residuos

## Librerías en uso

- Para ejecutar el presente código, estamos empleando las siguientes librerías:
  - `library(dplyr)` para manipular datos
  - `library(knitr)` para mostrar las tablas como texto plano en consola
  - `library(ggplot2)` para construir gráficos con más opciones
  - `library(texreg)` para reordenar los output de varios de modelos de regresión
  - `library(broom)` para extraer tablas de los output de los modelos de regresión
  - `library(lmtest)` para aplicar la prueba de Durbin Watson (correlación de residuales)

Para instalar estas librerías podemos ejecutar los siguientes códigos

```
-----  
#--  
# instalar librerias  
#--  
  
#--  
# librerias  
#--  
  
install.packages('tidyverse') # incluye a dplyr y ggplot2  
# junto a otras librerías útiles  
  
install.packages('knitr') # para mostrar tablas como texto plano  
  
install.packages('texreg') # para reordenar los output de varios  
# de modelos de regresión  
  
install.packages('broom') # para extraer tablas de los output  
# de los modelos de regresión  
  
install.packages('lmtest') # para evaluar la independencia  
# de las residuales
```

# Modelo de regresión con un predictor (bivariate regression)

Vik (2014) plantea, que podemos plantearnos tres preguntas generales acerca de dos variables:

- Q1: estan relacionadas estas dos variables?
- Q2: cuál es la dirección de la relación?
- Q3: que tan grande es la relación entre estas variables?

Para abordar estas preguntas empleando R, primero vamos a cargar los datos de la tabla 3.2. Luego de haber cargado estos datos, vamos a visualizar los datos mediante un dispersiograma o *scatter plot*. Finalmente, vamos a ajustar dos modelos de regresión: el modelo sin predictores (modelo nulo, o modelo compacto), y el modelo con predictores (modelo aumentado).

## Cargar datos Tabla 3.2

```
#-
# datos
#-

#-
# tabla 3.2
#-

data_table_3_2 <- read.table(
text="person y x x_q xy
1 2 8 64 16
2 3 9 81 27
3 3 9 81 27
4 4 10 100 40
5 7 6 36 42
6 5 7 49 35
7 5 4 16 20
8 7 5 25 35
9 8 3 9 24
10 9 1 1 9
11 9 2 4 18
12 10 2 4 20

",
header=TRUE, stringsAsFactors = FALSE)

#-
# mostrar tabla
#-

knitr::kable(data_table_3_2)
```

person	y	x	x_q	xy
1	2	8	64	16
2	3	9	81	27
3	3	9	81	27
4	4	10	100	40
5	7	6	36	42
6	5	7	49	35
7	5	4	16	20
8	7	5	25	35
9	8	3	9	24
10	9	1	1	9
11	9	2	4	18
12	10	2	4	20

Taller

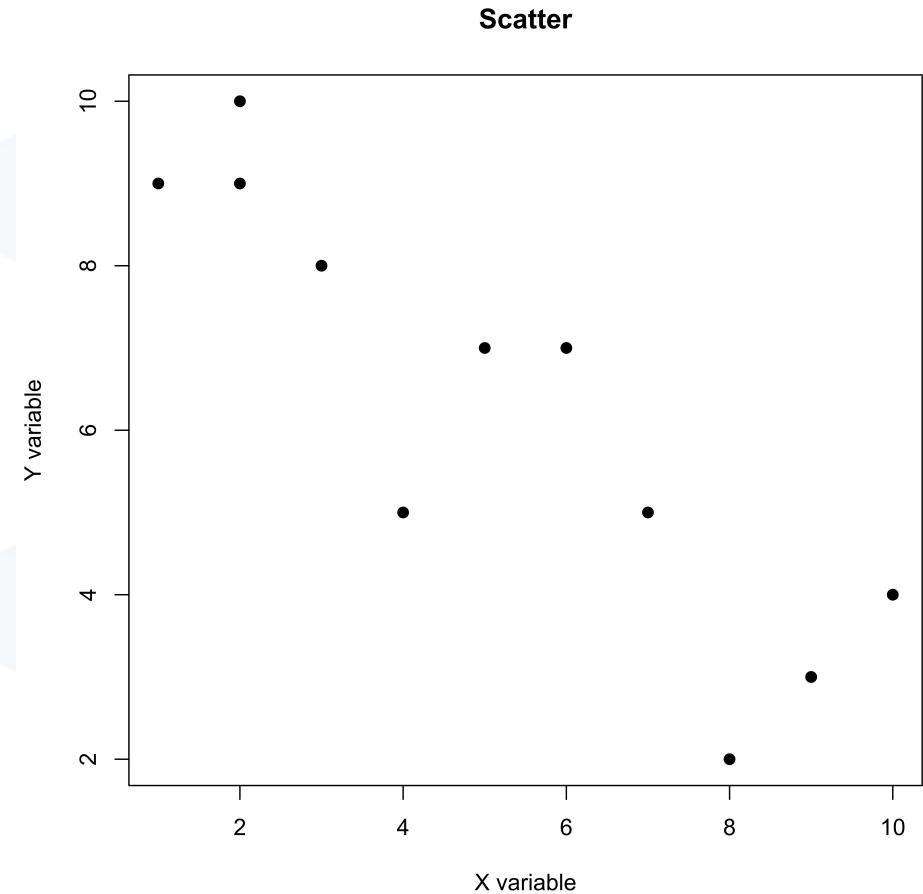
# Visualización de datos

Exploración gráfica de los datos

## Scatter plot tabla 3.2

```
#-----  
# datos  
#-----  
#-----  
# tabla 3.2  
#-----  
  
data_table_3_2 <- read.table(  
text="person y x x_q xy  
1 2 8 64 16  
2 3 9 81 27  
3 3 9 81 27  
4 4 10 100 40  
5 7 6 36 42  
6 5 7 49 35  
7 5 4 16 20  
8 7 5 25 35  
9 8 3 9 24  
10 9 1 1 9  
11 9 2 4 18  
12 10 2 4 20  
  
",  
header=TRUE, stringsAsFactors = FALSE)  
  
#-----  
# scatter  
#-----  
#-----  
# codigo base  
#-----  
  
with(data_table_3_2,  
plot(  
  x=x,  
  y=y,  
  xlab = "X variable",  
  ylab = "Y variable",  
  main = 'Scatter',  
  pch = 19,  
  frame = TRUE  
))
```

## Scatter con código base



# Scatter plot tabla 3.2

```
# datos
#-
#- tabla 3.2
#-

data_table_3_2 <- read.table(
text="person y x x_q xy
1 2 8 64 16
2 3 9 81 27
3 3 9 81 27
4 4 10 100 40
5 7 6 36 42
6 5 7 49 35
7 5 4 16 20
8 7 5 25 35
9 8 3 9 24
10 9 1 1 9
11 9 2 4 18
12 10 2 4 20

",
header=TRUE, stringsAsFactors = FALSE)

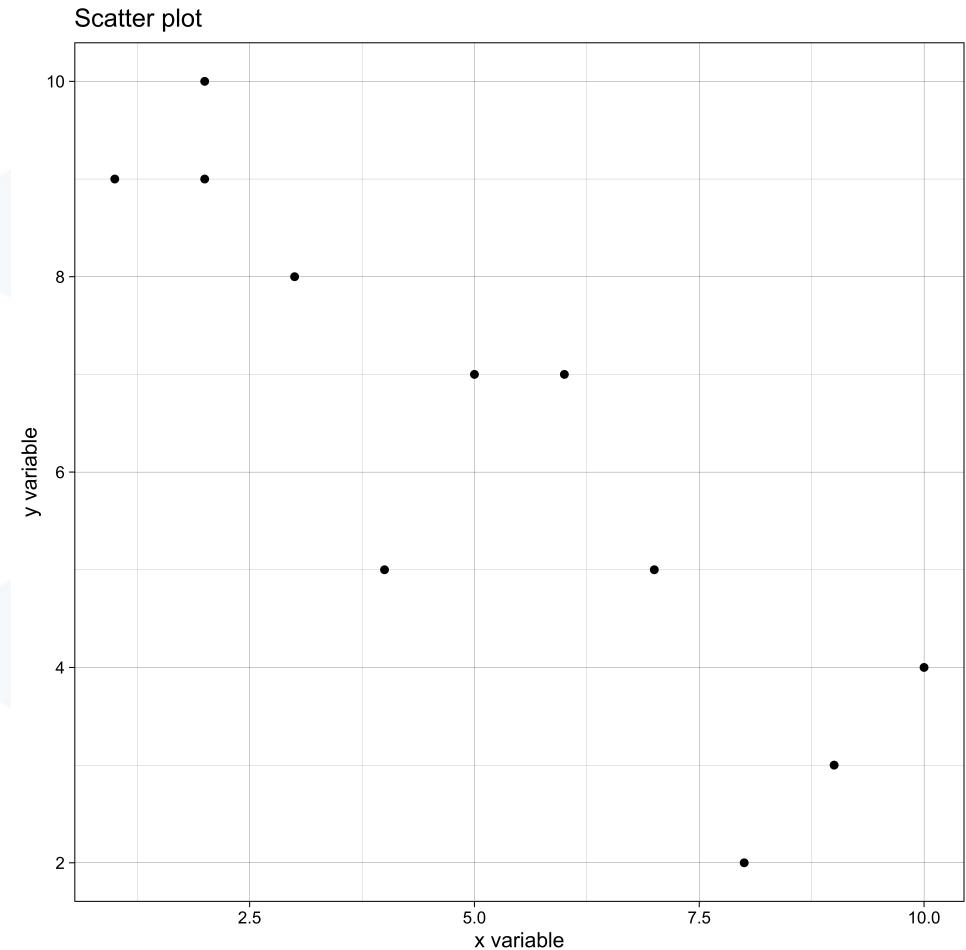
# scatter
#-

# codigo ggplot
#-

library(dplyr)
library(ggplot2)

data_table_3_2 %>%
ggplot(., aes(y = y, x = x)) +
geom_point(alpha = 1, col = 'black') +
xlab('x variable') +
ylab('y variable') +
ggtitle('Scatter plot') +
theme_linedraw()
```

# Scatter con ggplot2



Taller

# Ajuste de modelos

Especificación de modelos de regresión

# Ajustar modelo de regresión

Los modelos de regresión que vamos a ajustar son los siguientes:

Modelo Compacto o Modelo Nulo

$$y_i = \beta_0 + \epsilon_i$$

Modelo Aumentado, de un solo predictor

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Modelo Aumentado, con un predictor centrado a la gran media

$$y_i = \beta_0 + \beta_1(x_i - \bar{x}_i) + \epsilon_i$$

Para ajustar los modelos anteriores, primero vamos a preparar los datos. Vamos a centrar a la variable **X**, y vamos a crear la variable **x\_cgm**

```
#--#
# ajustar modelos
#--#
#--#
# preparar datos codigo base
#--#
data_model <- data_table_3_2[, c('y', 'x')]
data_model$x_cgm <- data_table_3_2$x - mean(data_table_3_2$x, na.rm = TRUE)

#--#
# preparar datos (dplyr)
#--#
data_model <- data_table_3_2 %>%
  mutate(x_g = mean(x, na.rm = TRUE)) %>%
  mutate(x_cgm = x - x_g) %>%
  dplyr::select(y, x, x_cgm) %>%
  dplyr::glimpse()
```

# Datos preparados

```
#--#
# mostrar datos
#--#
knitr::kable(data_model)
```

y	x	x_cgm
2	8	2.5
3	9	3.5
3	9	3.5
4	10	4.5
7	6	0.5
5	7	1.5
5	4	-1.5
7	5	-0.5
8	3	-2.5
9	1	-4.5
9	2	-3.5
10	2	-3.5

# Ajustar modelos

```
#--#
# formulas
#--#
f00 <- as.formula(y ~ + 1)
f01 <- as.formula(y ~ + 1 + x)
f02 <- as.formula(y ~ + 1 + x_cgm)

#--#
# ajustar modelos
#--#
m00 <- lm(f00, data = data_model)
m01 <- lm(f01, data = data_model)
m02 <- lm(f02, data = data_model)
```

# Resultados de los modelos ajustados

```
# ajustar modelos
#-----#
# formulas
#-----#
f00 <- as.formula(y ~ + 1)
f01 <- as.formula(y ~ + 1 + x)
f02 <- as.formula(y ~ + 1 + x_cgm)

#-----#
# ajustar modelos
#-----#
m00 <- lm(f00, data = data_model)
m01 <- lm(f01, data = data_model)
m02 <- lm(f02, data = data_model)

# comparar modelos de forma sintética
#-----#
texreg::screenreg(
  list(m00, m01, m02),
  star.symbol = "***",
  center = TRUE,
  doctype = FALSE,
  dcolumn = TRUE,
  booktabs = TRUE,
  single.row = FALSE
)

===== Model 1     Model 2     Model 3 =====
(Intercept) 6.00 *** 10.27 *** 6.00 ***
              (0.78)   (0.76)   (0.36)
x             -0.78 ***
                      (0.12)
x_cgm          -0.78 ***
                      (0.12)
----- R^2        0.00      0.80      0.80
Adj. R^2       0.00      0.79      0.79
Num. obs.      12        12        12
----- *** p < 0.001; ** p < 0.01; * p < 0.05
```

# Tamaños de efecto

```
#-----#
# tamaño de efecto de la relación
#-----

# como r de Pearson
with(data_model, cor(y, x))

[1] -0.8971007299

# como R2
summary(m02)$r.squared

[1] 0.8047897196

# R2 obtenido con library(broom)
broom::glance(m02)

# A tibble: 1 × 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik    AIC    BIC
  <dbl>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 0.805       0.785  1.25  41.2 0.0000763 1 -18.6 43.2 44.7
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

## Abordando las preguntas básicas de Vik

- Q1: estan relacionadas estas dos variables?
  - Sí, la variable **y** esta relacionada de forma lineal a la variable **x**.
- Q2: cuál es la dirección de la relación?
  - La forma de la relación es negativa.
  - A medida que aumentan los valores de **x**, esperamos que los valores **y** disminuyan.
- Q3: que tan grande es la relación entre estas variables?
  - La relación es grande. **x** e **y**, presentan una correlación alta ( $r = -.89$ ).
  - En términos de varianza explicada, **x** explica hasta un 80% de la varianza de **y** ( $R^2 = .80$ )

Taller

# Resultados de modelos de regresión

Inspección básica de los outputs de regresión

## Resultados del modelo nulo

```
#-----  
# resultados  
#-----  
  
#-----  
# coeficientes  
#-----  
  
summary(m00)  
  
Call:  
lm(formula = f00, data = data_model)  
  
Residuals:  
    Min      1Q Median      3Q     Max  
-4.00   -2.25    0.00    2.25    4.00  
  
Coefficients:  
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) 6.0000000  0.7784989 7.70714 0.0000092956 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 2.696799 on 11 degrees of freedom  
  
#-----  
# análisis de varianza  
#-----  
  
anova(m00)  
  
Analysis of Variance Table  
  
Response: y  
          Df Sum Sq Mean Sq F value Pr(>F)  
Residuals 11  80 7.2727273
```

## Resultados del modelo con un predictor

```
#-----  
# resultados  
#-----  
  
#-----  
# coeficientes  
#-----  
  
summary(m01)  
  
Call:  
lm(formula = f01, data = data_model)  
  
Residuals:  
    Min      1Q Median      3Q     Max  
-2.1635514 -0.3364486  0.1121495  0.7803738  1.4906542  
  
Coefficients:  
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) 10.2663551  0.7560713 13.57855 0.00000090657 ***  
x         -0.7757009  0.1208104 -6.42081 0.000076267145 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 1.249673 on 10 degrees of freedom  
Multiple R-squared:  0.8047897,   Adjusted R-squared:  0.7852687  
F-statistic: 41.22681 on 1 and 10 DF, p-value: 0.00007626715  
  
#-----  
# análisis de varianza  
#-----  
  
anova(m01)  
  
Analysis of Variance Table  
  
Response: y  
          Df Sum Sq Mean Sq F value Pr(>F)  
x           1 64.383178 64.383178 41.22681 0.000076267 ***  
Residuals 10 15.616822  1.561682  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Taller

# Interpretación de los modelos ajustados

Representación geométrica de los modelos ajustados

# Interpretación (m00)

- Intercepto ( $b_0$ )
- Coeficiente de regresión ( $b_1$ )

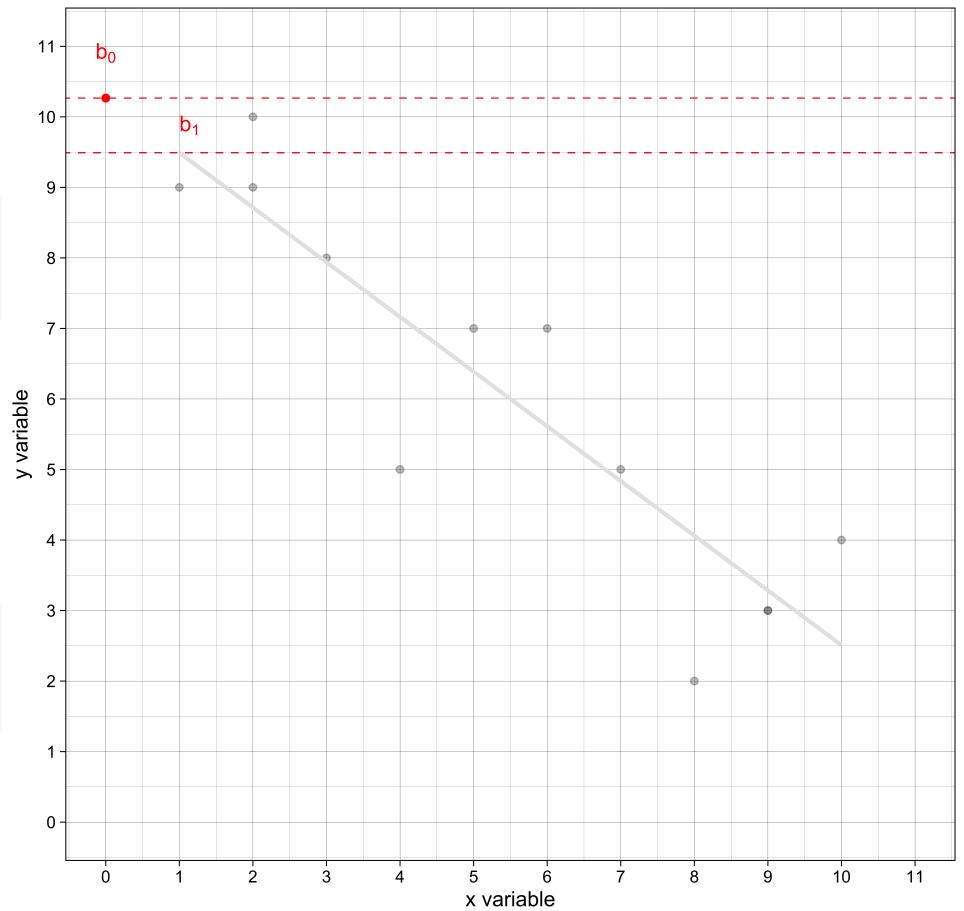
```
#-- proyección
#-----#
#-- intercepto
#-----#
b00_m01 <- lm(y ~ 1 + x, data = data_model) %>%
  broom::tidy() %>%
  dplyr::filter(term == '(Intercept)') %>%
  dplyr::select(estimate) %>%
  dplyr::pull()

b01_m01 <- lm(y ~ 1 + x, data = data_model) %>%
  broom::tidy() %>%
  dplyr::filter(term == 'x') %>%
  dplyr::select(estimate) %>%
  dplyr::pull()

#-- agregar valores esperados ( $y_{\hat{}}$ ), a una base de datos
#-----#
fitted_values <- lm(y ~ 1 + x, data = data_model) %>%
  predict()
data_plot_raw <- data_model %>%
  mutate(y_hat = fitted_values)

#-- proyección de modelo sobre datos observados
#-----#
data_plot_raw %>%
  ggplot(., aes(y = y, x = x)) +
  geom_point(alpha = .3, col = 'grey20') +
  geom_smooth(
    formula = y ~ x,
    method = 'lm',
    se = FALSE,
    colour = 'grey90',
    alpha = .005
  ) +
  # intercepto
  geom_point(aes(y = b00_m01, x = 0), col = 'red', alpha = .5) +
  # intercepto
  geom_abline(
    intercept = b00_m01,
    slope = 0,
    color="red",
    linetype="dashed",
    size=.3
  ) +
  annotate('text',
    x = 0,
    y = b00_m01 + .5,
    size = 4,
    vjust = 0,
    colour = 'red',
    label = paste0('b[0]'),
    parse = TRUE
  )
  # b01_m01
  geom_abline(
    intercept = b00_m01 + b01_m01,
    slope = 0,
    color="red",
    linetype="dashed",
    size=.3
  ) +
  annotate('text',
    x = 1,
    y = b00_m01 + (b01_m01)/2,
    size = 4,
    hjust = 0,
    colour = 'red',
    label = paste0('b[01]'),
    parse = TRUE
  )
  xlab('x variable') +
  ylab('y variable') +
  scale_x_continuous(breaks=seq(0,11,1), limits = c(0, 11)) +
  scale_y_continuous(breaks=seq(0,11,1), limits = c(0, 11)) +
  labs(title = expression(y[i] == b[0] + b[1] * x[i] + epsilon[i]),
       parse = TRUE)
```

$$y_i = b_0 + b_1 x_i + \varepsilon_i$$



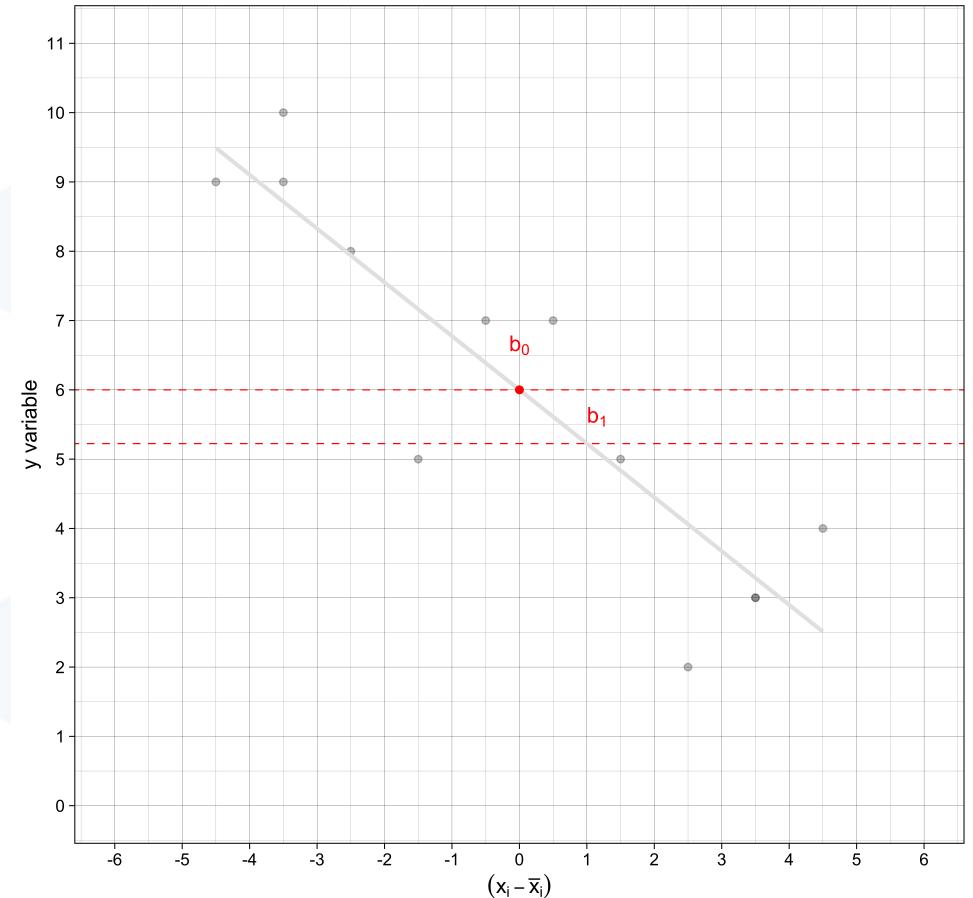
# Interpretación (m01)

- Intercepto ( $b_0$ ), cuando  $x_i - \bar{x}_i$  es el predictor
- Coeficiente de regresión ( $b_1$ )

```
#-- proyección
#-- intercepto
#-- agregar valores esperados (y_hat), a una base de datos
#-- proyección de modelo sobre datos observados
#-- punto del intercepto
#-- beta 1

#-- centrado de variable
xlab(expression((x[i] - bar(x)[i]))) +
ylab("y variable") +
scale_y_continuous(breaks=seq(0,11,1), limits = c(0, 11)) +
scale_x_continuous(breaks=seq(-6,6,1), limits = c(-6, 6)) +
labs(
title = expression(y[i] == b[0] + b[1] * (x[i] - bar(x)[i]) + epsilon[i]),
parse = TRUE
)
```

$$y_i = b_0 + b_1(x_i - \bar{x}_i) + \epsilon_i$$



Nota: El centrado de variable es útil para darle una interpretación sustantiva al intercepto del modelo ajustado.

# Interpretación (m01)

## Interpretación de resultados

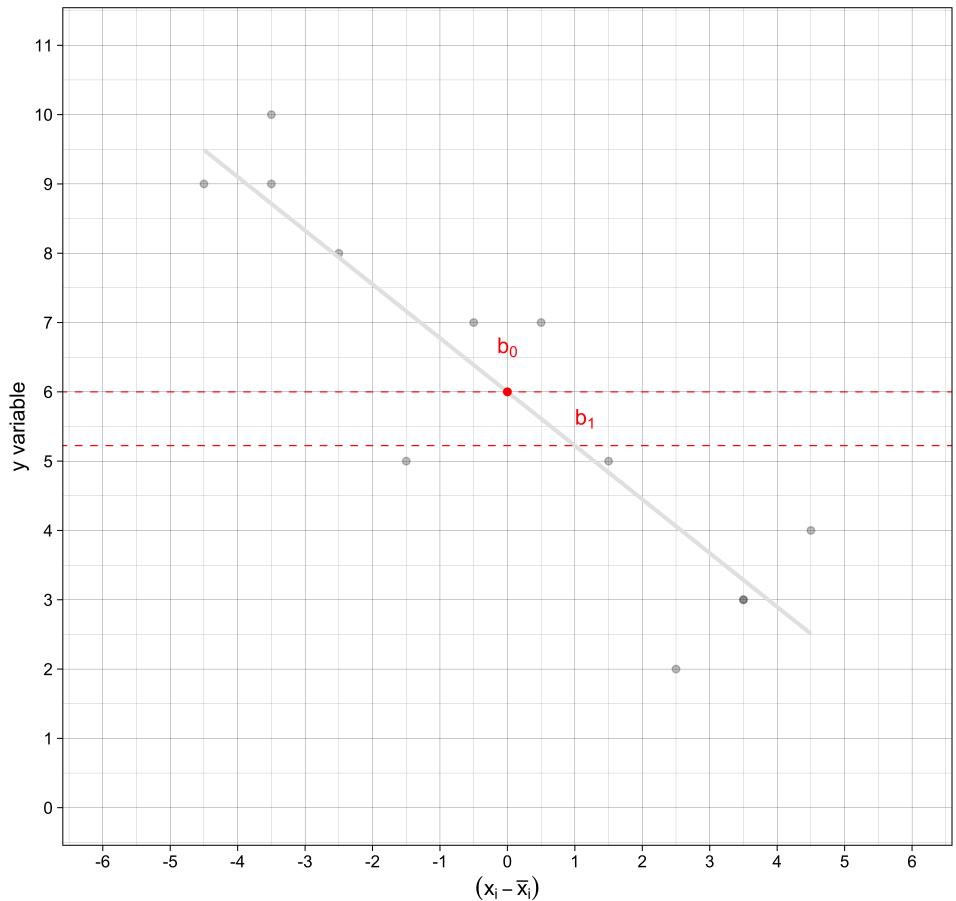
### Intercepto ( $b_0$ )

- $b_0$  corresponde a la media de  $y_i$ , cuando  $x_i$  es ingresado al modelo con sus valores centrados.
- Otra forma de interpretar este estimado, es que  $b_0$  es el valor esperado de  $y_i$ , cuando  $(x_i - \bar{x}_i) = 0$ .
- En otras palabras,  $b_0$  es el valor que esperamos de  $y_i$ , a valores promedio de  $x_i$ .

### Coeficiente ( $b_1$ )

- $b_1$  corresponde al valor que esperamos que tome  $y_i$ , condicional al cambio de una unidad de  $x_i$ .
- Empleando los resultados del modelo ajustado (m02), esperamos que los casos donde  $(x_i - \bar{x}_i) = 1$ , obtengan  $b_0 + b_1$  valores en la escala de  $y_i$ .

$$y_i = b_0 + b_1(x_i - \bar{x}_i) + \varepsilon_i$$



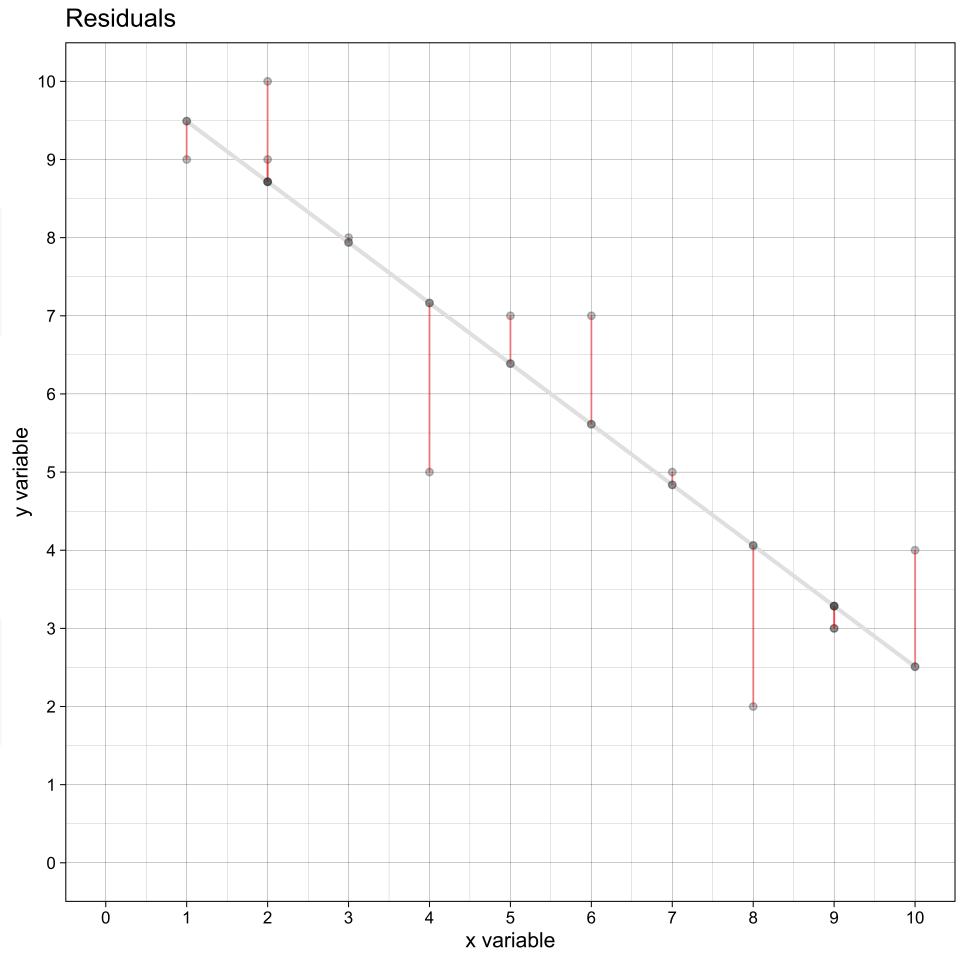
Taller

# Visualización de errores

Representación geométrica de los residuales del modelo

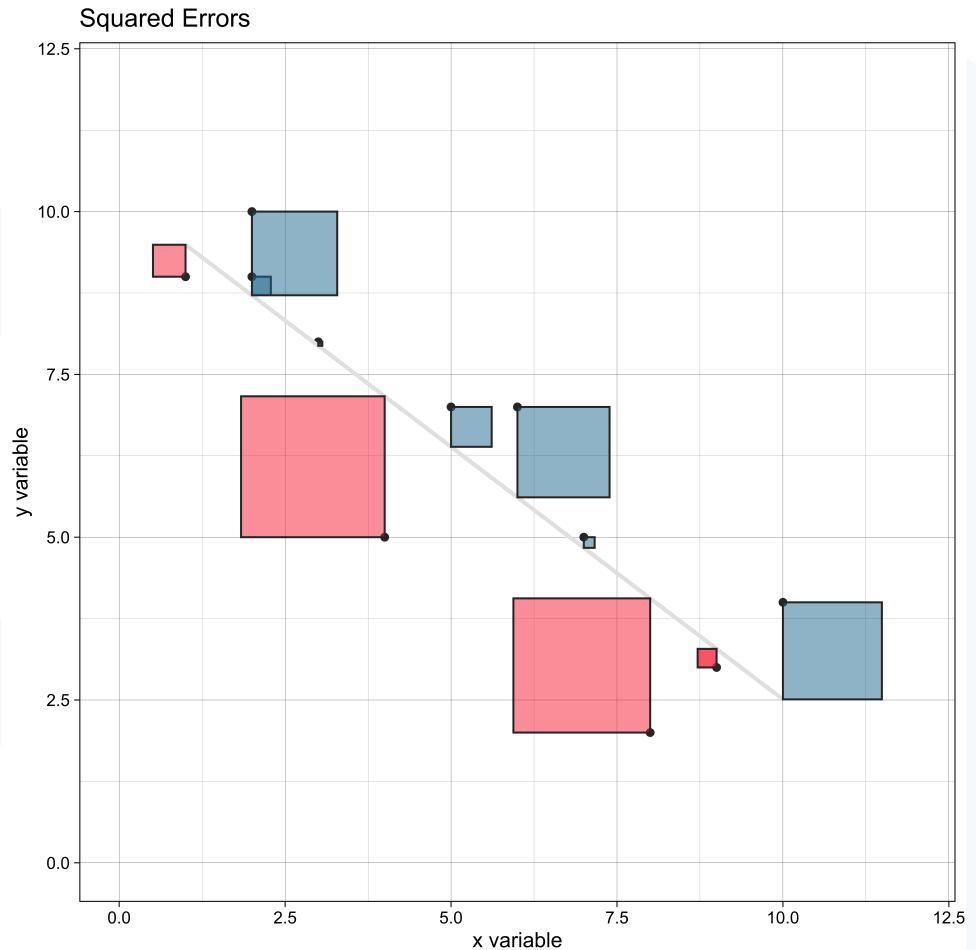
## Lollipop plot de Residuales

```
#-----  
# residuales  
#-----  
  
#-----  
# crear valores esperados  
#-----  
  
fitted_values <- lm(y ~ + 1 + x, data = data_model) %>%  
  predict()  
  
#-----  
# agregar valores esperados ( $y_{\hat{}}$ ), a una base de datos  
#-----  
  
data_fitted <- data_model %>%  
  mutate(y_hat = fitted_values)  
  
#-----  
# lollipop plot  
#-----  
  
data_fitted %>%  
  ggplot(., aes(y = y, x = x)) +  
  geom_point(alpha = .3, col = 'grey20') +  
  geom_smooth(  
    formula = y ~ x,  
    method = "lm",  
    se = FALSE,  
    colour = "grey90",  
    alpha = .005  
  ) +  
  geom_segment(aes(xend = x, yend = y_hat), alpha = .4, col = 'red') +  
  geom_point(aes(y = y_hat), col = 'grey25', alpha = .5) +  
  xlab('x variable') +  
  ylab('y variable') +  
  scale_x_continuous(breaks=seq(0,10,1), limits = c(0, 10)) +  
  scale_y_continuous(breaks=seq(0,10,1), limits = c(0, 10)) +  
  ggtitle('Residuals') +  
  theme_linedraw()
```



## Squared errors

```
#-----  
# errores cuadrados  
#-----  
  
#-----  
# crear valores esperados  
#-----  
  
fitted_values <- lm(y ~ + 1 + x, data = data_model) %>%  
  predict()  
  
#-----  
# agregar valores esperados (y_hat), a una base de datos  
#-----  
  
data_fitted <- data_model %>%  
  mutate(y_hat = fitted_values) %>%  
  mutate(res = y - y_hat) %>%  
  mutate(res_colour = case_when(  
    res == 0 ~ 'grey80',  
    res > 0 ~ '#247BA0', # red  
    res < 0 ~ '#FB3640' # blue  
  ))  
  
rect_colour <- data_fitted$res_colour  
  
#-----  
# squared errors  
#-----  
  
data_fitted %>%  
  ggplot(., aes(y = y, x = x)) +  
  geom_point(alpha = 1, col = 'grey20') +  
  ylim(0,12) +  
  xlim(0,12) +  
  geom_smooth(  
    formula = y ~ x,  
    method = "lm",  
    se = FALSE,  
    colour = "grey90",  
    alpha = .005  
  ) +  
  geom_rect(  
    aes(  
      xmin=x,  
      xmax=x+res,  
      ymin=y-res,  
      ymax=y  
    ),  
    colour = 'grey20',  
    fill=rect_colour,  
    alpha=0.5  
  ) +  
  xlab('x variable') +  
  ylab('y variable') +  
  gtitle('Squared Errors') +  
  theme_linedraw()
```



Taller

# Error total del modelo

Cómo obtener el error de cada modelo ajustado

# Error por modelo

## Sumas de errores cuadrados

La suma de errores cuadrados consiste en la suma total de residuales de un modelo. En otras palabras, es la suma de todas las distancias cuadráticas entre los valores observados, y los valores esperados por un modelo.

$$SSE = \sum (y_i - \hat{y}_i)^2$$

En Vik (2014) se emplea más de una formula para obtener las sumas de cuadrados, diferenciando entre el modelo nulo o modelo compacto, y el modelo aumentado. Por conveniencia, vamos a crear una función que nos permita obtener la suma de errores de cuadrados de cada modelo ajustado.

```
#-----#
# errores cuadrados
#-----#
#-----#
# función SSE
#-----#
sse_model <- function(model){
  y      <- model$model[,1]
  y_hat <- model$fitted.values
  squared_errors <- (y - y_hat)^2
  sse <- sum(squared_errors)
  return(sse)
}
```

```
#-----
# extraer SSE
#-----
sse_00 <- sse_model(m00)
sse_01 <- sse_model(m01)
sse_02 <- sse_model(m02)

#-----
# crear tabla
#-----

sse_table <- data.frame(
  model = c(
    'null model',
    'augmented model',
    'centered x model'
  ),
  see = c(sse_00, sse_01, sse_02)
)
#-----
# mostrar tabla
#-----
knitr::kable(sse_table, digits = 2)
```

model	see
null model	80.00
augmented model	15.62
centered x model	15.62

## Descomponer Error total

El error total del modelo nulo (o modelo compacto), puede ser descompuesto en términos del error reducido (o explicado), y lo que nos queda (el residuo). Esta idea puede ser expresada de la siguiente forma:

$$SST = SSR + SSE_{m01}$$

**SST** es el error total, que lo obtenemos del modelo compacto o modelo nulo. El error reducido **SSR**, o error explicado lo obtenemos como la resta entre el error total, menos el error residual del modelo aumentado (ver Vik, 2014, p24). Y finalmente, el error residual **SSE<sub>m01</sub>** son los errores cuadrados del modelo aumentado.

```
#-----#
# des compositon de errores
#-----#
#-----#
# obtener medidas de error
#-----#
total_error <- sse_00
residual_error <- sse_01
explained_error <- total_error - residual_error

#-----#
# crear tabla
#-----#
error_table <- data.frame(
  error = c(
    'Total_Error',
    'Residual_Error',
    'Explained_Error'
  ),
  value = c(
    total_error,
    residual_error,
    explained_error
  )
)
```

```
#-----#
# mostrar tabla
#-----#
knitr::kable(error_table, digits = 2)



| error           | value |
|-----------------|-------|
| Total_Error     | 80.00 |
| Residual_Error  | 15.62 |
| Explained_Error | 64.38 |


#-----#
# proporcion de varianza explicada
#-----#
# proporcion de SSE sobre el total
explained_error/total_error

[1] 0.8047897196

# R2 del modelo
summary(m01)$r.squared

[1] 0.8047897196
```

Taller

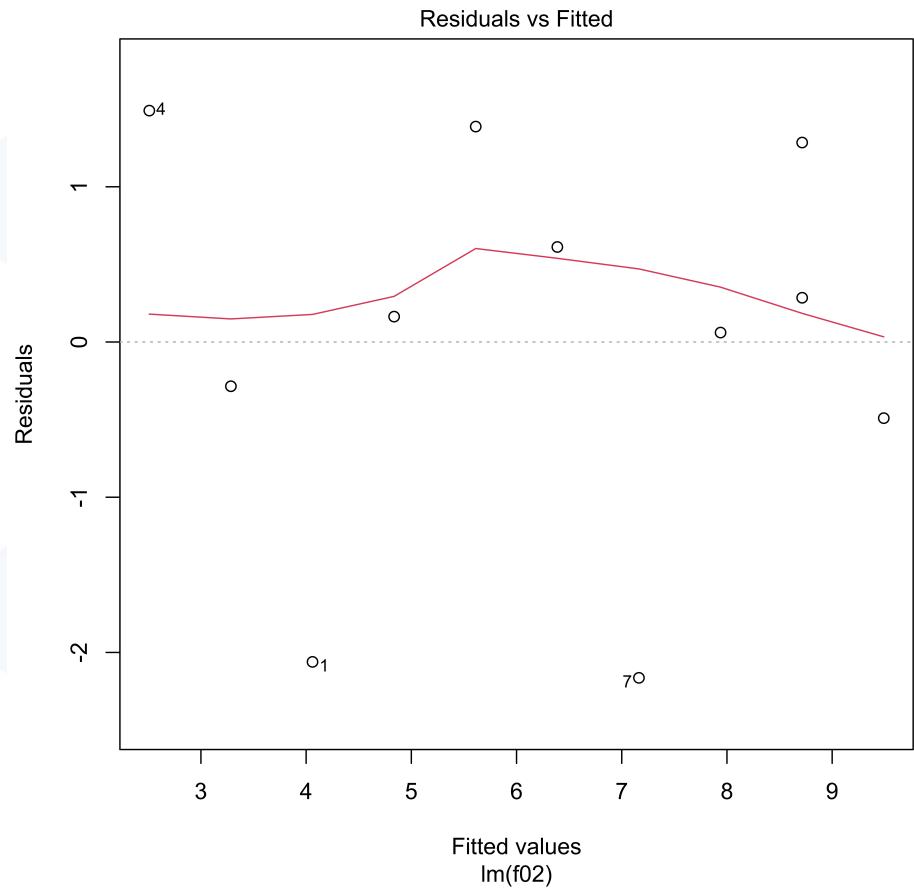
# Evaluación de supuestos de la regresión

Linealidad, Homocedasticidad, Normalidad de los residuales e Independencia de los residuales

## Linealidad del modelo

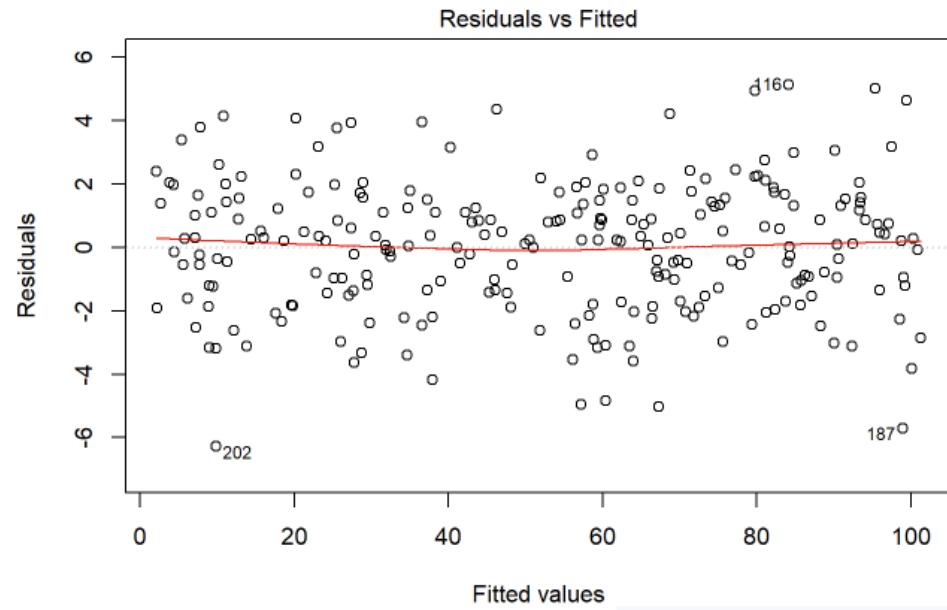
```
#-----  
# diagnósticos  
#-----  
# residual vs fitted  
#-----  
plot(m02, 1)
```

- Lo que esperamos, es que la dispersión de nuestros residuos sea homogénea al rededor de los datos ajustados.
- Esperamos, que nuestra linea central en este gráfico sea horizontal.

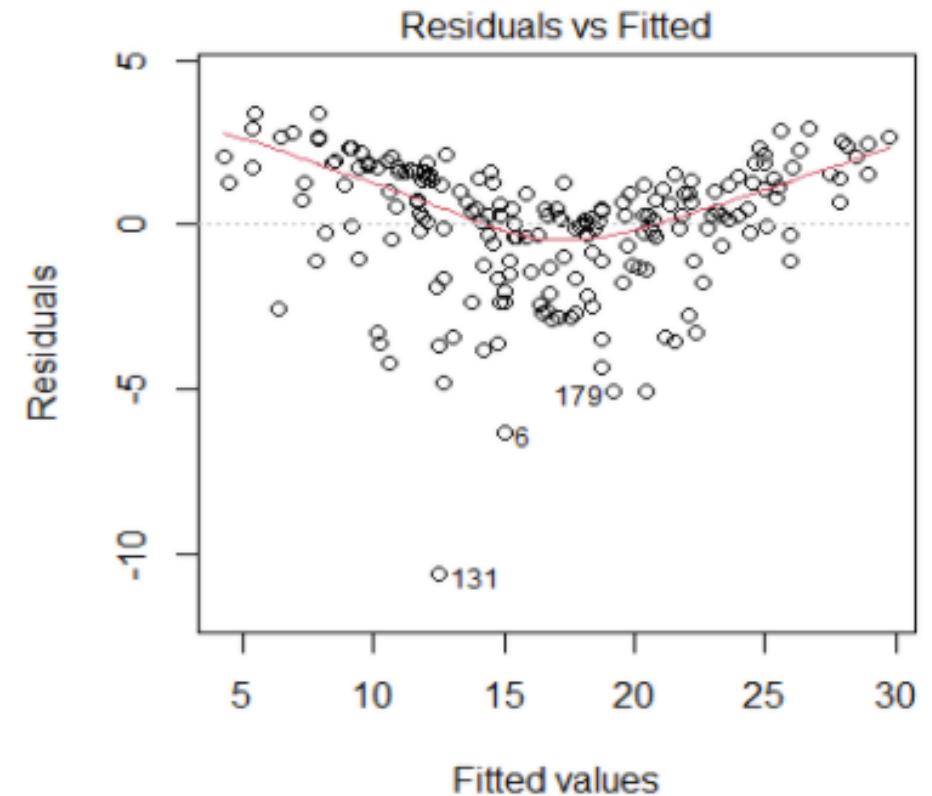


## Linealidad del modelo

### Linealidad ideal



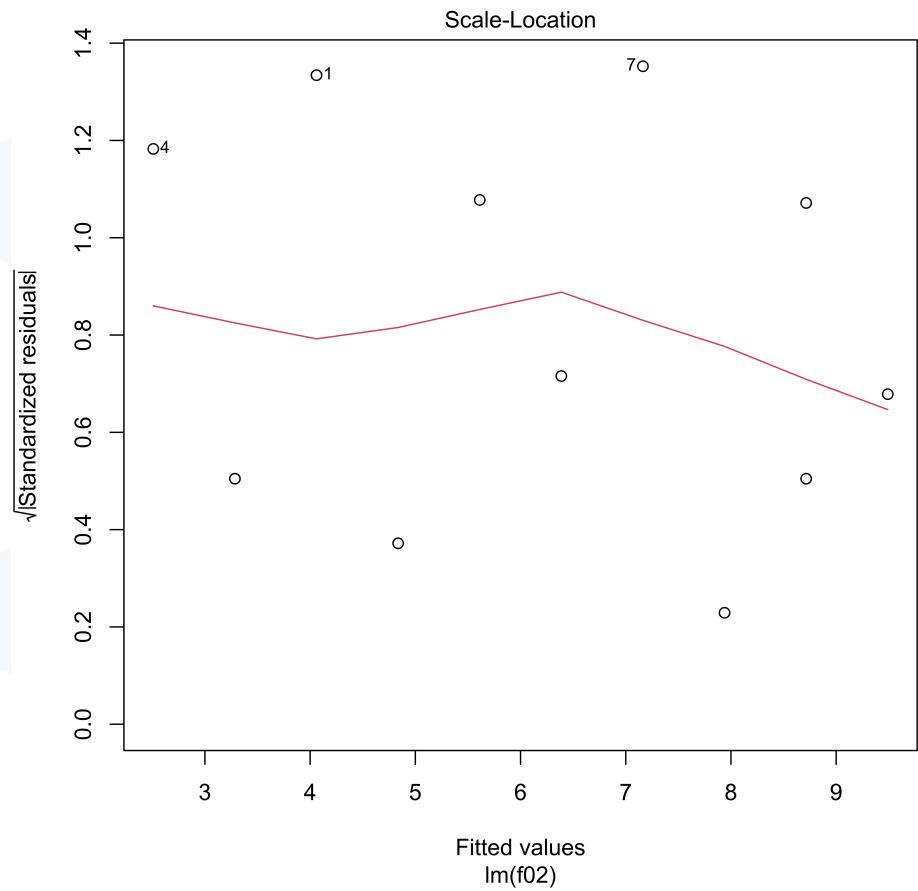
### Supuesto de linealidad no cumplido



## Homocedasticidad

```
#-----  
# diagnósticos  
#-----  
  
#-----  
# homocedasticity  
#-----  
plot(m02, 3)  
  
#-----  
# homocedasticity test  
#-----  
  
# Breusch-Pagan test  
lmtest::bttest(m02)
```

- La homocedasticidad refiere a que observemos varianzas similares de los errores del modelo (los residuales), a diferentes valores predichos
- Esperamos que, **nuestros residuales tomen posiciones similares a lo largo de largo de los valores ajustados.**
- La prueba Breusch-Pagan evalua si los residuales presentan varianza similar a todos los valores predichos, o si esta difiere (no hay varianza homogenea).
  - Los resultados de la aprueba no apoyan la hipotesis de que la varianza de los residuales fuera heterocedastica a los valores esperados por el modelo.



## Homocedasticidad

```
#-----  
# diagnósticos  
#-----  
  
#-----  
# homocedasticity  
#-----  
  
plot(m02, 3)  
  
#-----  
# homocedasticity test  
#-----  
  
# Breusch-Pagan test  
lmtest::bptest(m02)
```

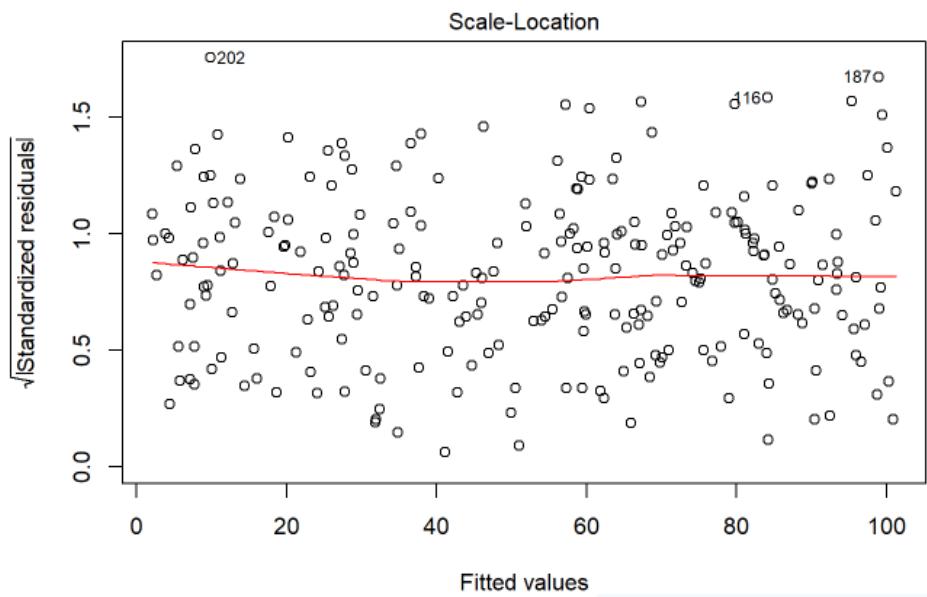
studentized Breusch-Pagan test

data: m02  
BP = 0.21984401, df = 1, p-value = 0.6391588

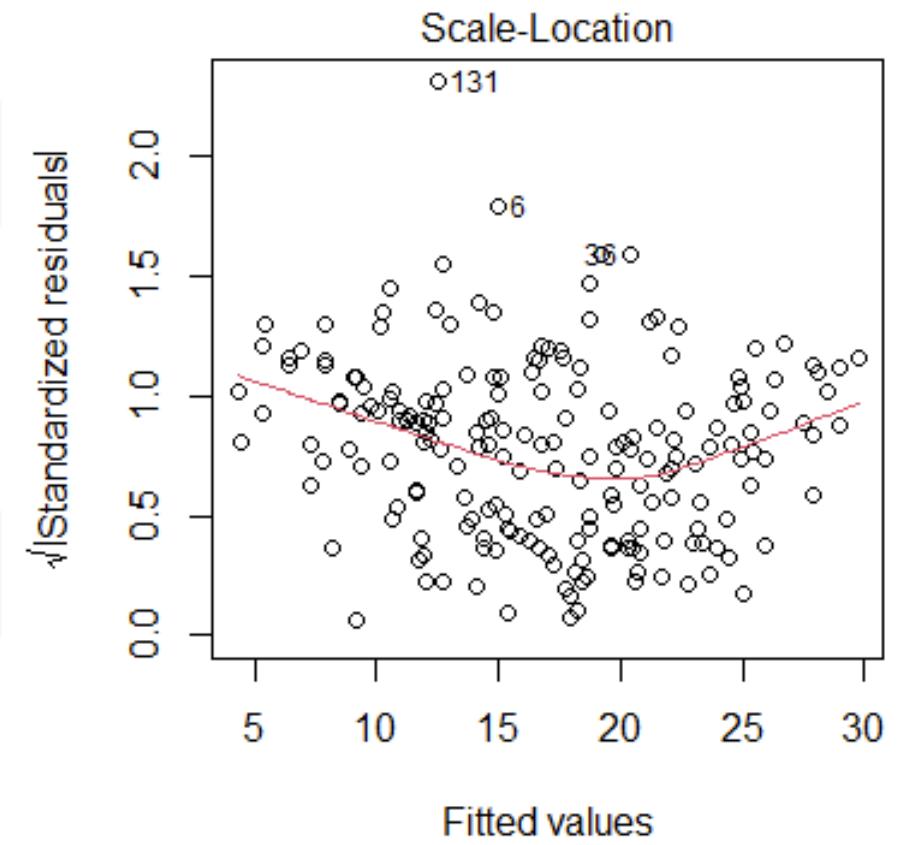
- La homocedasticidad refiere a que observemos varianzas similares de los errores del modelo (los residuales), a diferentes valores predichos
- Esperamos que, nuestros residuales tomen posiciones similares a lo largo de largo de los valores ajustados.
- **La prueba Breusch-Pagan evalua si los residuales presentan varianza similar a todos los valores predichos, o si esta difiere (no hay varianza homogena).**
  - Los resultados de la aprueba no apoyan la hipotesis de que la varianza de los residuales fuera heterocedastica a los valores esperados por el modelo.

## Homocedasticidad

Homocedasticidad ideal



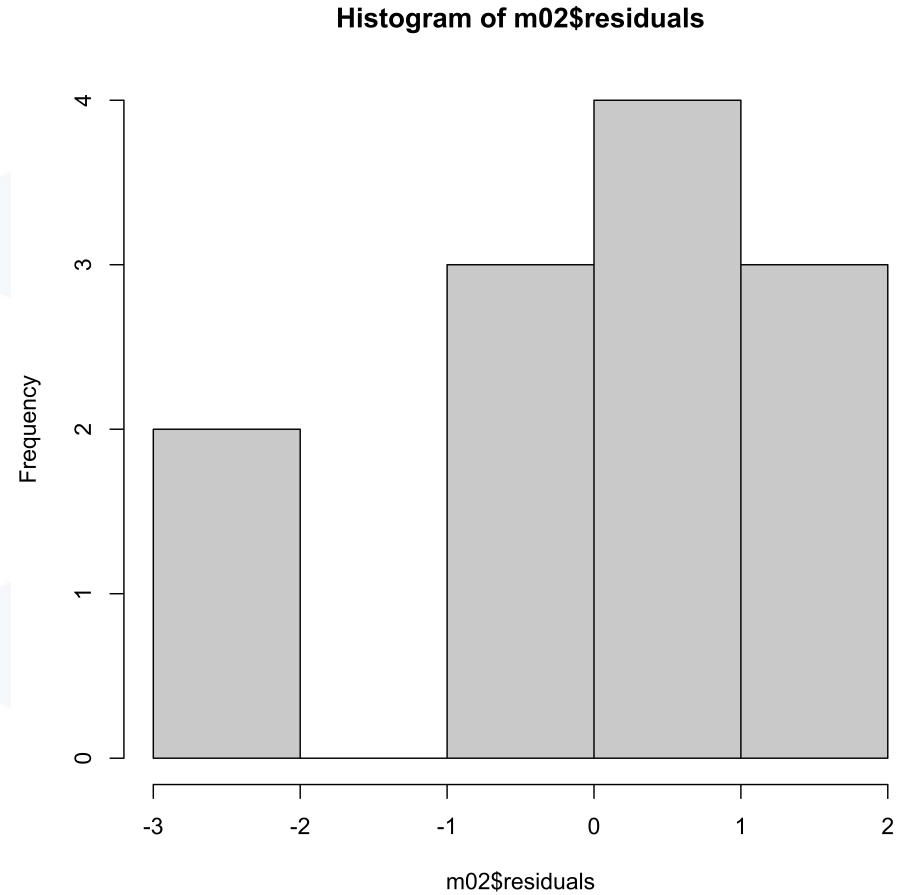
Supuesto de Homocedasticidad no cumplido



## Normalidad

```
#-----  
# diagnósticos  
#-----  
#-----  
# histograma de residuos  
#-----  
hist(m02$residuals)  
#-----  
# normalidad de los residuos  
#-----  
plot(m02, 2)  
#-----  
# Komogorov Smirnoff test  
#-----  
ks.test(m02$residuals,  
  "pnorm",  
  mean=mean(m02$residuals),  
  sd=sd(m02$residuals))
```

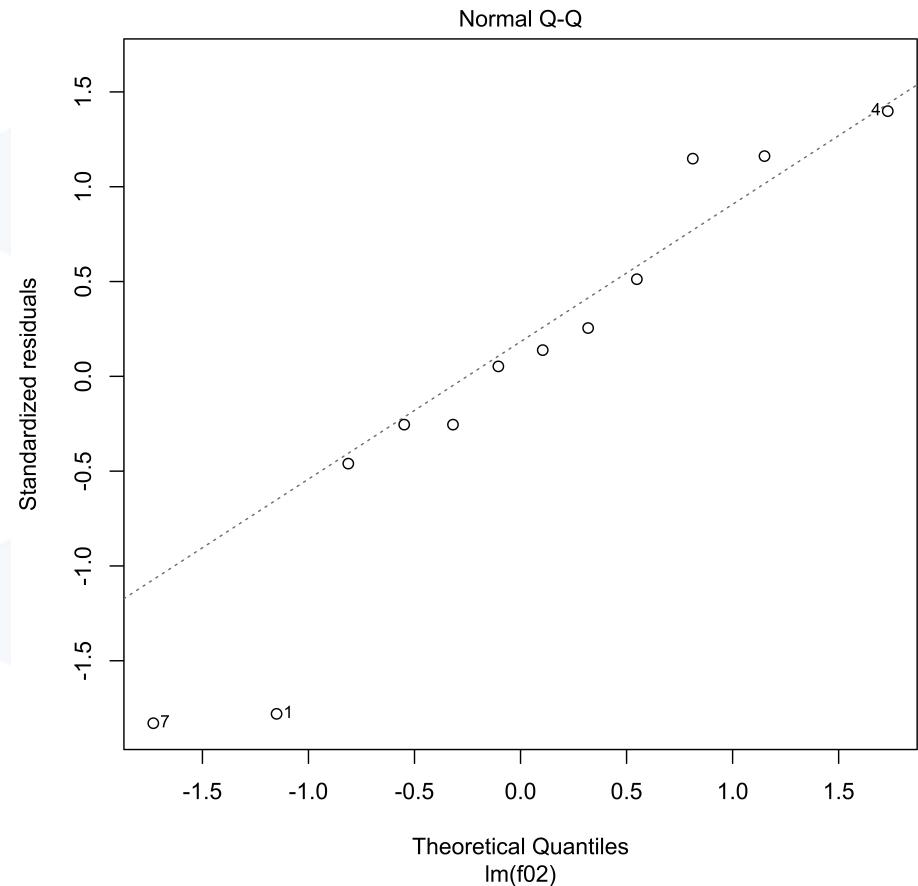
- Uno modelo que ajusta bien a los datos, debiera presentar residuales que presenten una distribución normal.
- Los QQ plots, ordenan a los residuos en una diagonal, de tal forma que si estos se "desalinean" podemos sospechar que el supuesto de normalidad de los errores no se cumple.
- Podemos emplear el test Kolmogorov Smirnoff, y evaluar si la distribucion de errores discrepa de una distribución normal.
  - En este caso, los resultados de la prueba aplicada indican que nuestros datos no discrepan de la distribución esperada.



## Normalidad

```
#-----  
# diagnósticos  
#-----  
  
#-----  
# histograma de residuos  
#-----  
  
hist(m02$residuals)  
#-----  
# normalidad de los residuos  
#-----  
  
plot(m02, 2)  
  
#-----  
# Komogorov Smirnoff test  
#-----  
  
ks.test(m02$residuals,  
        "pnorm",  
        mean=mean(m02$residuals),  
        sd=sd(m02$residuals))
```

- Uno modelo que ajusta bien a los datos, debiera presentar residuales que presenten una distribución normal.
- Los QQ plots, ordenan a los residuos en una diagonal, de tal forma que si estos se "desalinean" podemos sospechar que el supuesto de normalidad de los errores no se cumple.
- Podemos emplear el test Kolmogorov Smirnoff, y evaluar si la distribución de errores discrepa de una distribución normal.
  - En este caso, los resultados de la prueba aplicada indican que nuestros datos no discrepan de la distribución esperada.



# Normalidad

```
#-----  
# diagnósticos  
#-----  
  
#-----  
# histograma de residuos  
#-----  
  
hist(m02$residuals)  
  
#-----  
# normalidad de los residuos  
#-----  
  
plot(m02, 2)  
  
#-----  
# Komogorov Smirnoff test  
#-----  
  
ks.test(m02$residuals,  
  "pnorm",  
  mean=mean(m02$residuals),  
  sd=sd(m02$residuals)  
)
```

One-sample Kolmogorov-Smirnov test

```
data: m02$residuals  
D = 0.1735801, p-value = 0.8045496  
alternative hypothesis: two-sided
```

- Uno modelo que ajusta bien a los datos, debiera presentar residuales que presenten una distribución normal.
- Los QQ plots, ordenan a los residuos en una diagonal, de tal forma que si estos se "desalinean" podemos sospechar que el supuesto de normalidad de los errores no se cumple.
- **Podemos emplear el test Kolmogorov Smirnoff, y evaluar si la distribucion de errores discrepa de una distribución normal.**
  - En este caso, los resultados de la prueba aplicada indican que nuestros datos no discrepan de la distribución esperada.

## Independencia de los errores

```
#-----  
# diagnósticos  
#-----  
#-----  
# independencia de los errores  
#-----  
lmtest::dwtest(m02, alternative = 'two.sided')
```

- Esperamos que la auto correlación entre los residuos sea casi nula
- La prueba de Durbin-Watson evalua si hay correlaciones entre pares de residuales adjacentes (Field et al., 2012).
- Los resultados del Durbin-Watson test, indican que nuestros residuales no presentan autocorrelaciones
  - Empleamos una prueba de *dos colas*, asumiendo que los residuales incluso podrian presentar *autocorrelaciones negativas* (ver Long & Teetor, 2019, p376).
  - En caso de que pudiera defenderse, que solo son razonables las correlaciones positivas, el test podria aplicarse con una sola cola.

Durbin-Watson test

```
data: m02  
DW = 1.482083, p-value = 0.1917543  
alternative hypothesis: true autocorrelation is not 0
```

# Independencia de los errores

## Supuesto de independencia de los errores sin cumplir

```
#-- diagnósticos  
#--  
# # caso de observaciones anidadas  
#--  
data_nested <- readRDS('nld_16.rds')  
#--  
# Durbin Watson test  
#--  
  
lm(civ ~ ses, data = data_nested) %>%  
  lmtest::dwtest(., alternative = 'two.sided')  
  
Durbin-Watson test  
  
data: .  
DW = 1.3248382, p-value < 0.000000000000022204  
alternative hypothesis: true autocorrelation is not 0
```

Nota: cuando se ignora a las escuelas en el modelo, el factor de anidación, encontramos evidencias de que los residuales no son independientes entre sí.

## Supuesto de independencia de los errores satisfecho

```
#--  
# diagnósticos  
#--  
#--  
# # caso de observaciones anidadas  
#--  
data_nested <- readRDS('nld_16.rds')  
#--  
# Durbin Watson test  
#--  
  
lm(civ ~ ses + as.factor(id_j), data = data_nested) %>%  
  lmtest::dwtest(., alternative = 'two.sided')
```

```
Durbin-Watson test  
  
data: .  
DW = 2.1111814, p-value = 0.5275726  
alternative hypothesis: true autocorrelation is not 0
```

Nota: se cumple el supuesto de independencia, en caso de que se sature la varianza entre escuelas, incluyendo a las escuelas como efectos fijos.

# Muchas gracias!

# Referencias

Long, J. D., & Teator, P. (2019). R Cookbook. O'Reilly.

Vik, P. (2014). Regression, ANOVA, and the general linear model: A statistics primer. Sage.

Field, A., Miles, J., & Field, Z. (2012). Discovering Statistics using R. SAGE Publications Ltd.