# Temario de la sesión

1. Regresión lineal
   1.1. Modelo lineal simple en R
   1.2. Diagnóstico inicial del modelo
   1.3. Criterios de selección de un modelo
   1.4. Verificación de supuestos y pruebas de bondad de ajuste
2. Procesos Estocasticos en R
   2.1. Cadenas de Markov
   2.2. Processo Poisson (simple, compuesto y no-homogeneo)
   2.3. Teoría de colas
3. Aplicaciones y extras
   3.1. Paquetes actuariales en R (lifecontingencies, actuar) *(opcional)*
   3.2. Uso de ChatGPT como apoyo en programación y estadística en R *(opcional)*

# 1. Regresión lineal

La regresión lineal es uno de los modelos estadísticos más utilizados para explicar la relación entre una variable respuesta $Y$ y una o más variables explicativas $X$. El objetivo es:

- Modelar cómo cambia $Y$ en función de $X$.
- Predecir el valor de $Y$ para nuevas observaciones de $X$.
- Interpretar el efecto de los regresores sobre la respuesta.

El modelo lineal parte de la forma general:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \varepsilon_i$$

donde:

- $\beta_0$ es la ordenada al origen (intercepto).
- $\beta_j$ mide el cambio esperado en $Y$ por un cambio unitario en $X_j$.
- $\varepsilon_i \sim N(0, \sigma^2)$ son los errores aleatorios.

En esta sección veremos primero el caso más simple con una sola variable explicativa (regresión simple), para luego extendernos a varios regresores y a temas de diagnóstico y selección de modelos.

## 1.1. Modelo lineal simple en R

El modelo lineal simple tiene la forma:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

- $\beta_0$: intercepto, valor esperado de $Y$ cuando $X = 0$.
- $\beta_1$: pendiente, cambio esperado en $Y$ por unidad de $X$.

El ajuste en R se realiza con la función `lm()` (linear model).

**Ejemplo en R: datos simulados**

```
set.seed(123)
x <- 1:20
y <- 5 + 2*x + rnorm(20, mean = 0, sd = 3)  # modelo con ruido

modelo <- lm(y ~ x)
summary(modelo)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.964 -1.808 -0.113  1.558  5.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.9303     1.3860   4.279 0.000452 ***
```

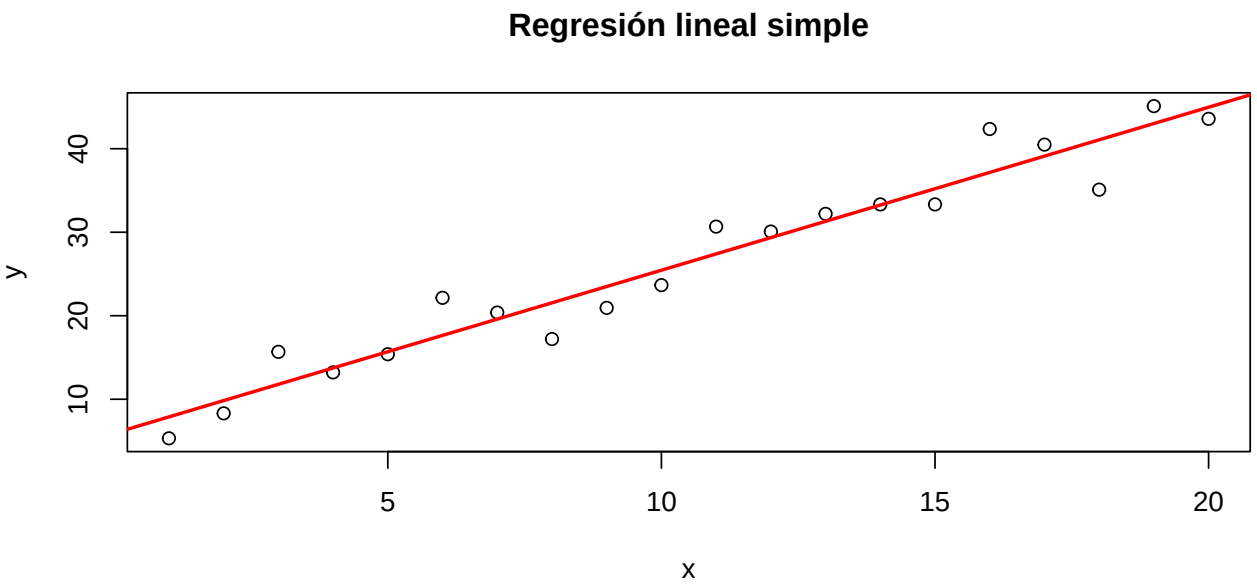
```
## x            1.9519     0.1157  16.870 1.78e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.984 on 18 degrees of freedom
## Multiple R-squared:  0.9405, Adjusted R-squared:  0.9372
## F-statistic: 284.6 on 1 and 18 DF,  p-value: 1.778e-12
```

Interpretación:

- El output muestra estimaciones de $\hat{\beta}_0$ y $\hat{\beta}_1$.
- El summary() incluye errores estándar, valor t, p-value y $R^2$.
- En este caso, la pendiente estimada debe estar cerca de 2.

### Visualización

```r
plot(x, y, main = "Regresión lineal simple")
abline(modelo, col = "red", lwd = 2)
```




Esto muestra la nube de puntos y la recta ajustada.

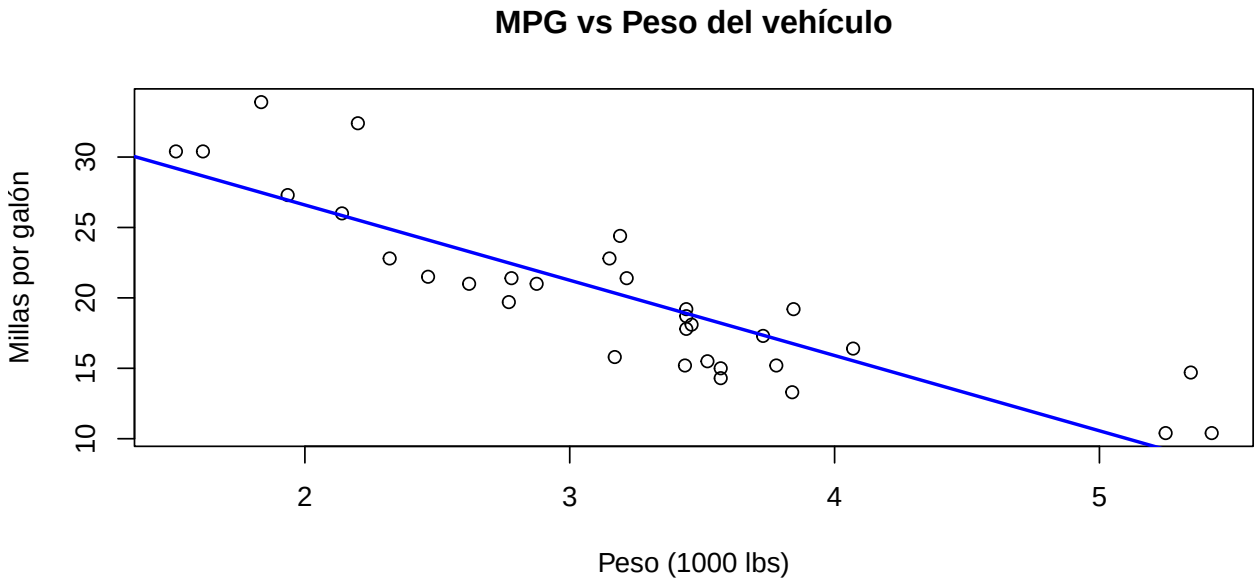
### Ejemplo en R: dataset real (mtcars)

```r
# Relación entre peso (wt) y consumo de combustible (mpg)
modelo2 <- lm(mpg ~ wt, data = mtcars)
summary(modelo2)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
```

```
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```r
plot(mtcars$wt, mtcars$mpg, main = "MPG vs Peso del vehículo",
     xlab = "Peso (1000 lbs)", ylab = "Millas por galón")
abline(modelo2, col = "blue", lwd = 2)
```




Interpretación:

- Se observa una relación negativa: a mayor peso, menor rendimiento de combustible.
- El coeficiente de pendiente $\hat{\beta}_1$ indica cuántos mpg se pierden por cada 1000 libras extra de peso.

## 1.2. Diagnóstico y selección del modelo

El modelo lineal clásico asume que los errores $\varepsilon_i$:

1. Tienen media cero:

$$\mathbb{E}[\varepsilon_i] = 0$$

2. Son homocedásticos (varianza constante):

$$\text{Var}(\varepsilon_i) = \sigma^2$$

3. Son independientes:

$$\text{Cov}(\varepsilon_i, \varepsilon_j) = 0 \quad \text{para} \quad i \neq j$$

   No hay correlación entre observaciones.

4. Son normales:

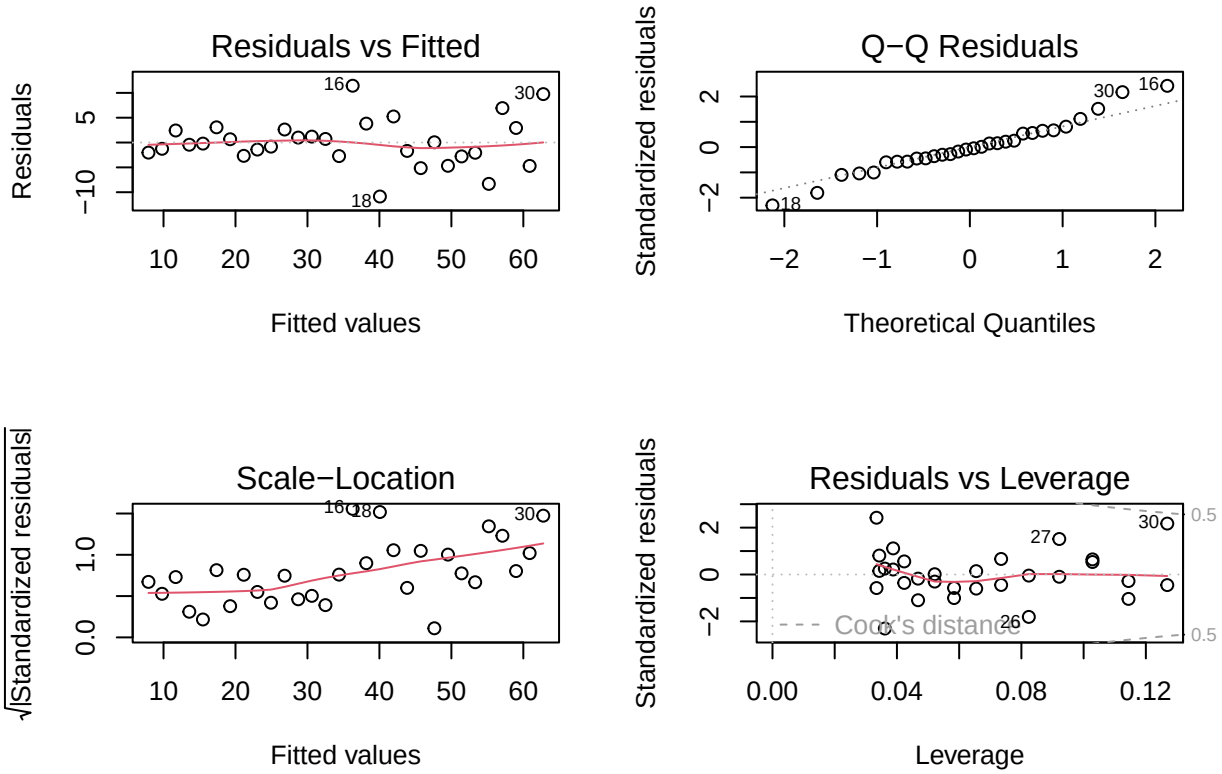
$$\varepsilon_i \sim N(0, \sigma^2).$$

**Herramientas de diagnóstico en R**

1) Gráficos de residuales

```
set.seed(123)
x <- 1:30
y <- 5 + 2*x + rnorm(30, sd = ifelse(x < 15, 2, 6))  # heterocedasticidad simulada
modelo <- lm(y ~ x)

par(mfrow=c(2,2))
plot(modelo)  # residuales, QQ-plot, leverage, Cook's distance
```




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

- Residuals vs Fitted: debe verse nube aleatoria (detecta no-linealidad o heterocedasticidad).
- Normal Q–Q: residuales deben alinearse en la diagonal (normalidad).
- Scale-Location: detecta heterocedasticidad (debe ser horizontal).
- Residuals vs Leverage: identifica puntos influyentes (Cook's distance).

2) Prueba de homocedasticidad (Breusch–Pagan)

```
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
bptest(modelo)  # H0: varianza constante
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo
## BP = 4.7819, df = 1, p-value = 0.02876
```

- si el valor-p es bajo, es evidencia de heterocedasticidad.

3) Prueba de autocorrelación (Durbin–Watson)

```r
dwtest(modelo)  # H0: no autocorrelación
```

```
##
##  Durbin-Watson test
##
## data:  modelo
## DW = 2.2928, p-value = 0.7309
## alternative hypothesis: true autocorrelation is greater than 0
```

- si el valor-p es bajo, los residuales están correlacionados (problema típico en series de tiempo).

4) Multicolinealidad (VIF)

Se usa en modelos con múltiples regresores.

```r
library(car)
```

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

```r
modelo_multi <- lm(mpg ~ wt + hp + disp, data = mtcars)
vif(modelo_multi)
```

```
##       wt       hp     disp
## 4.844618 2.736633 7.324517
```

- si el VIF $> 10$, implica multicolinealidad severa.

5) Prueba de normalidad de residuales

```r
shapiro.test(resid(modelo))  # H0: residuales ~ N(0,varianza cte)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(modelo)
## W = 0.97319, p-value = 0.6297
```

- si tengo un valor-p bajo, implica que los residuales no son normales.

## 1.3. Selección de un modelo

Una vez verificados los supuestos básicos del modelo lineal, surge la pregunta: ¿Qué variables deben incluirse en el modelo?

- Un modelo demasiado pequeño (subespecificado) omite variables relevantes, introduciendo sesgo.
- Un modelo demasiado grande (sobreespecificado) aumenta la varianza de los estimadores y puede dificultar la interpretación.

La selección de modelos busca un equilibrio entre ajuste y parsimonia.

**Criterios estadísticos comunes:**

- $R^2$ y $R^2$ ajustado:
  - El $R^2$ siempre crece al añadir variables.
  - El $R^2_{\text{ajustado}}$ penaliza la complejidad y puede disminuir si las variables añadidas no aportan valor.
- $C_p$ de Mallows:
  - Compara un modelo reducido con el completo.
  - Un buen modelo cumple $C_p \approx p + 1$, donde $p$ es el número de regresores.
- AIC (Akaike Information Criterion) y BIC (Bayesian Information Criterion):
  - Valores más bajos indican mejor balance ajuste–complejidad.
- PRESS (Prediction Error Sum of Squares) o validación cruzada:
  - Evalúan la capacidad predictiva fuera de la muestra.

Métodos computacionales

- Selección exhaustiva: evaluar los $2^k$ modelos posibles (factible solo con pocos regresores).
- Selección paso a paso (stepwise):
  - Forward: empieza vacío y agrega variables.
  - Backward: empieza completo y elimina variables.
  - Mixto: combina ambos criterios.

**Ejemplo en R: selección paso a paso con AIC**

```r
library(MASS)

# Datos de ejemplo: consumo de combustible en mtcars
modelo_completo <- lm(mpg ~ ., data = mtcars)

# Selección backward (parte del modelo completo)
modelo_step <- stepAIC(modelo_completo, direction = "backward")
```

```
## Start:  AIC=70.9
## mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - cyl   1    0.0799 147.57 68.915
## - vs    1    0.1601 147.66 68.932
## - carb  1    0.4067 147.90 68.986
## - gear  1    1.3531 148.85 69.190
```

```
## - drat  1    1.6270 149.12 69.249
## - disp  1    3.9167 151.41 69.736
## - hp    1    6.8399 154.33 70.348
## - qsec  1    8.8641 156.36 70.765
## <none>              147.49 70.898
## - am    1   10.5467 158.04 71.108
## - wt    1   27.0144 174.51 74.280
##
## Step:  AIC=68.92
## mpg ~ disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - vs    1    0.2685 147.84 66.973
## - carb  1    0.5201 148.09 67.028
## - gear  1    1.8211 149.40 67.308
## - drat  1    1.9826 149.56 67.342
## - disp  1    3.9009 151.47 67.750
## - hp    1    7.3632 154.94 68.473
## <none>              147.57 68.915
## - qsec  1   10.0933 157.67 69.032
## - am    1   11.8359 159.41 69.384
## - wt    1   27.0280 174.60 72.297
##
## Step:  AIC=66.97
## mpg ~ disp + hp + drat + wt + qsec + am + gear + carb
##
##        Df Sum of Sq    RSS    AIC
## - carb  1    0.6855 148.53 65.121
## - gear  1    2.1437 149.99 65.434
## - drat  1    2.2139 150.06 65.449
## - disp  1    3.6467 151.49 65.753
## - hp    1    7.1060 154.95 66.475
## <none>              147.84 66.973
## - am    1   11.5694 159.41 67.384
## - qsec  1   15.6830 163.53 68.200
## - wt    1   27.3799 175.22 70.410
##
## Step:  AIC=65.12
## mpg ~ disp + hp + drat + wt + qsec + am + gear
##
##        Df Sum of Sq    RSS    AIC
## - gear  1     1.565 150.09 63.457
## - drat  1     1.932 150.46 63.535
## <none>              148.53 65.121
## - disp  1    10.110 158.64 65.229
## - am    1    12.323 160.85 65.672
## - hp    1    14.826 163.35 66.166
## - qsec  1    26.408 174.94 68.358
## - wt    1    69.127 217.66 75.350
##
## Step:  AIC=63.46
## mpg ~ disp + hp + drat + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - drat  1     3.345 153.44 62.162
## - disp  1     8.545 158.64 63.229
## <none>              150.09 63.457
## - hp    1    13.285 163.38 64.171
```

```
## - am    1    20.036 170.13 65.466
## - qsec  1    25.574 175.67 66.491
## - wt    1    67.572 217.66 73.351
##
## Step:  AIC=62.16
## mpg ~ disp + hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - disp  1     6.629 160.07 61.515
## <none>              153.44 62.162
## - hp    1    12.572 166.01 62.682
## - qsec  1    26.470 179.91 65.255
## - am    1    32.198 185.63 66.258
## - wt    1    69.043 222.48 72.051
##
## Step:  AIC=61.52
## mpg ~ hp + wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## - hp    1     9.219 169.29 61.307
## <none>              160.07 61.515
## - qsec  1    20.225 180.29 63.323
## - am    1    25.993 186.06 64.331
## - wt    1    78.494 238.56 72.284
##
## Step:  AIC=61.31
## mpg ~ wt + qsec + am
##
##        Df Sum of Sq    RSS    AIC
## <none>              169.29 61.307
## - am    1    26.178 195.46 63.908
## - qsec  1   109.034 278.32 75.217
## - wt    1   183.347 352.63 82.790
```

```
summary(modelo_step)
```

```
##
## Call:
## lm(formula = mpg ~ wt + qsec + am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4811 -1.5555 -0.7257  1.4110  4.6610
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.6178     6.9596   1.382 0.177915
## wt           -3.9165     0.7112  -5.507 6.95e-06 ***
## qsec          1.2259     0.2887   4.247 0.000216 ***
## am            2.9358     1.4109   2.081 0.046716 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.459 on 28 degrees of freedom
## Multiple R-squared:  0.8497, Adjusted R-squared:  0.8336
## F-statistic: 52.75 on 3 and 28 DF,  p-value: 1.21e-11
```

Interpretación:

- stepAIC elimina variables que no mejoran el AIC.
- El summary() del modelo final muestra las variables retenidas y su significancia.
- Puedes comparar el AIC y el $R^2$ ajustado del modelo final contra el completo.

## 1.4. Verificación de supuestos

El ajuste de un modelo de regresión no garantiza su validez. Es indispensable verificar los supuestos clásicos y, si se violan, aplicar correcciones o transformaciones.

1. Supuesto de normalidad

- Gráfico: QQ-plot de residuales.
- Pruebas: Shapiro–Wilk, Kolmogorov–Smirnov, Jarque–Bera.

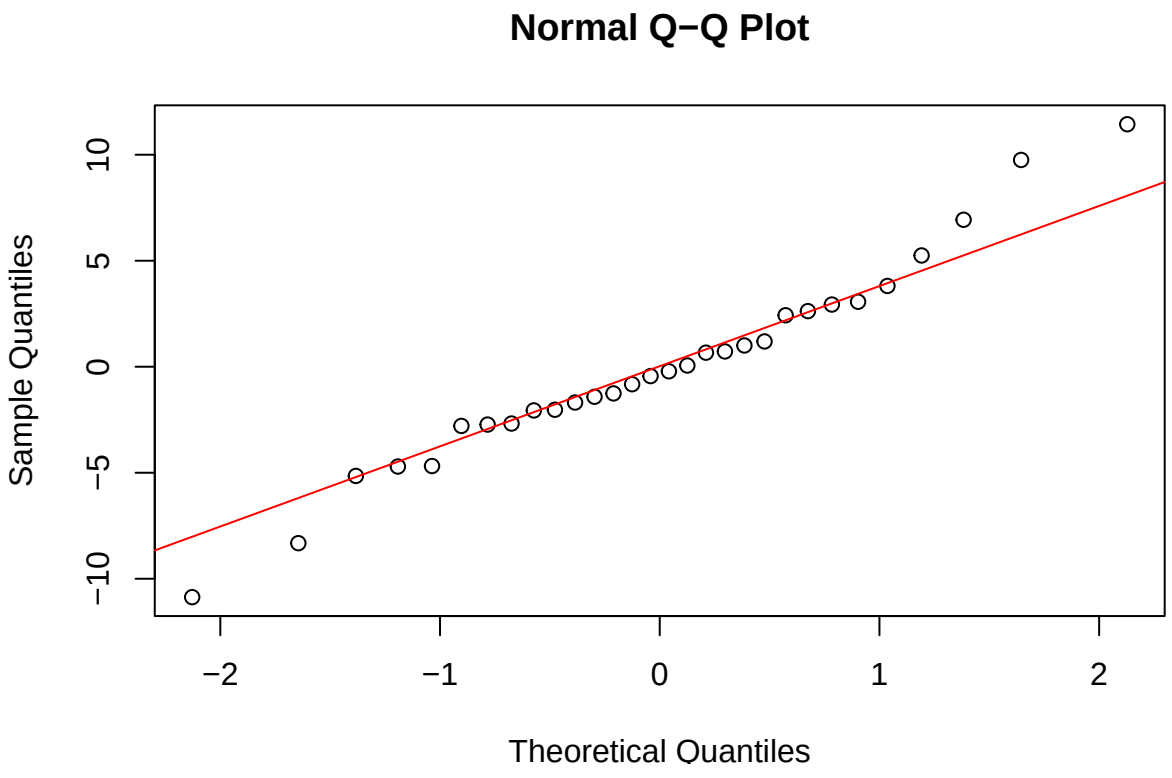
```
shapiro.test(resid(modelo))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(modelo)
## W = 0.97319, p-value = 0.6297
```

```
ks.test(resid(modelo), "pnorm", mean=0, sd=sd(resid(modelo)))
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  resid(modelo)
## D = 0.11043, p-value = 0.8191
## alternative hypothesis: two-sided
```

```
qqnorm(resid(modelo)); qqline(resid(modelo), col="red")
```



2. Supuesto de homocedasticidad

- Gráfico: Residuales vs ajustados (esperamos nube horizontal).
- Prueba formal: Breusch–Pagan (bptest), White test.

```
library(lmtest)
bptest(modelo)   # H0: varianza constante
```

```
##
##  studentized Breusch-Pagan test
##
## data:  modelo
## BP = 4.7819, df = 1, p-value = 0.02876
```

3. Supuesto de independencia

- Series de tiempo: revisar autocorrelación en los residuales (ACF, PACF).
- Prueba formal: Durbin–Watson (dwtest).

```
dwtest(modelo)   # H0: no autocorrelación
```

```
##
##  Durbin-Watson test
##
## data:  modelo
## DW = 2.2928, p-value = 0.7309
## alternative hypothesis: true autocorrelation is greater than 0
```

4. Supuesto de ausencia de multicolinealidad

- Diagnóstico: factores de inflación de varianza (VIF).

```
library(car)
vif(modelo_multi)
```

```
##       wt       hp     disp
## 4.844618 2.736633 7.324517
```

5. Bondad de ajuste

```
y <- rpois(100, lambda = 3)
obs <- table(factor(y, levels=0:8))
esp <- dpois(0:8, lambda=3) * length(y)
chisq.test(obs, p=esp/sum(esp))
```

```
## Warning in chisq.test(obs, p = esp/sum(esp)): Chi-squared approximation may be
## incorrect
```

```
##
##  Chi-squared test for given probabilities
##
## data:  obs
## X-squared = 5.426, df = 8, p-value = 0.7112
```

# 2. Procesos Estocasticos en R

Un proceso estocástico es una colección de variables aleatorias indexadas por el tiempo (discreto o continuo) que describe la evolución de un sistema incierto. Estos modelos son fundamentales en estadística, finanzas, actuaría y ciencias de datos, pues permiten:

- Modelar fenómenos dependientes del tiempo (colas, inventarios, series financieras).
- Simular sistemas aleatorios con memoria (dependencia entre estados).
- Calcular distribuciones de probabilidades asociadas a eventos complejos.

En esta sección veremos algunos de los procesos más importantes y cómo implementarlos en R:

- Cadenas de Markov (procesos discretos sin memoria).
- Algoritmo Metropolis–Hastings (simulación de distribuciones con MCMC).
- Procesos de Poisson (modelan conteos en tiempo).
- Teoría de colas (aplicaciones en servicios e infraestructura).

## 2.1. Cadenas de Markov

Una Cadena de Markov es un proceso estocástico discreto con la propiedad de Markov:

$$\Pr(X_{n+1} = j \mid X_n = i, X_{n-1}, \ldots, X_0) = \Pr(X_{n+1} = j \mid X_n = i)$$

Es decir, el futuro solo depende del presente, no del pasado completo.

Componentes clave

- Espacio de estados: conjunto de valores posibles $\{1, 2, \ldots, m\}$.
- Matriz de transición $P = [p_{ij}]$:

$$p_{ij} = \Pr(X_{n+1} = j \mid X_n = i), \quad \sum_j p_{ij} = 1$$

**Ejemplo en R: simulación de una cadena**

```
set.seed(123)

# Definimos matriz de transición
P <- matrix(c(0.7,0.3,
              0.4,0.6), byrow=TRUE, nrow=2)
colnames(P) <- rownames(P) <- c("A","B")
P
```

```
##     A   B
## A 0.7 0.3
## B 0.4 0.6
```

```
# Simulación de la cadena
n <- 20
X <- character(n)
X[1] <- "A"  # estado inicial

for (t in 2:n) {
  X[t] <- sample(c("A","B"), size=1, prob=P[X[t-1],])
}

X
```

```
##  [1] "A" "A" "B" "B" "A" "B" "B" "B" "A" "A" "A" "B" "B" "A" "A" "A" "B" "B" "B"
## [20] "B"
```

```r
table(X)/n  # frecuencias empíricas
```

```
## X
##    A    B
## 0.45 0.55
```

**Ejemplo: cálculo de distribuciones a $n$ pasos**

La distribución de estados después de $n$ pasos se obtiene con:

$$\pi^{(n)} = \pi^{(0)} P^n$$

donde $\pi^{(0)}$ es la distribución inicial.

```r
library(expm)
```

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

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```r
P <- matrix(c(0.7,0.3,
              0.4,0.6), byrow=TRUE, nrow=2)

pi0 <- c(1,0)    # distribución inicial

pi5 <- pi0 %*% (P %^% 5)   # P elevado a la 5
pi5
```

```
##         [,1]    [,2]
## [1,] 0.57247 0.42753
```

## 2.2. Processo Poisson (simple, compuesto y no-homogeneo)

El Proceso de Poisson es uno de los procesos estocásticos más usados en probabilidad aplicada, seguros, colas y fenómenos de conteo. Modela la ocurrencia aleatoria de eventos en el tiempo (o en el espacio), bajo ciertas condiciones.

1) Proceso de Poisson simple (homogéneo)

Definición: Un proceso $\{N(t), t \geq 0\}$ es un Proceso de Poisson de tasa $\lambda > 0$ si:

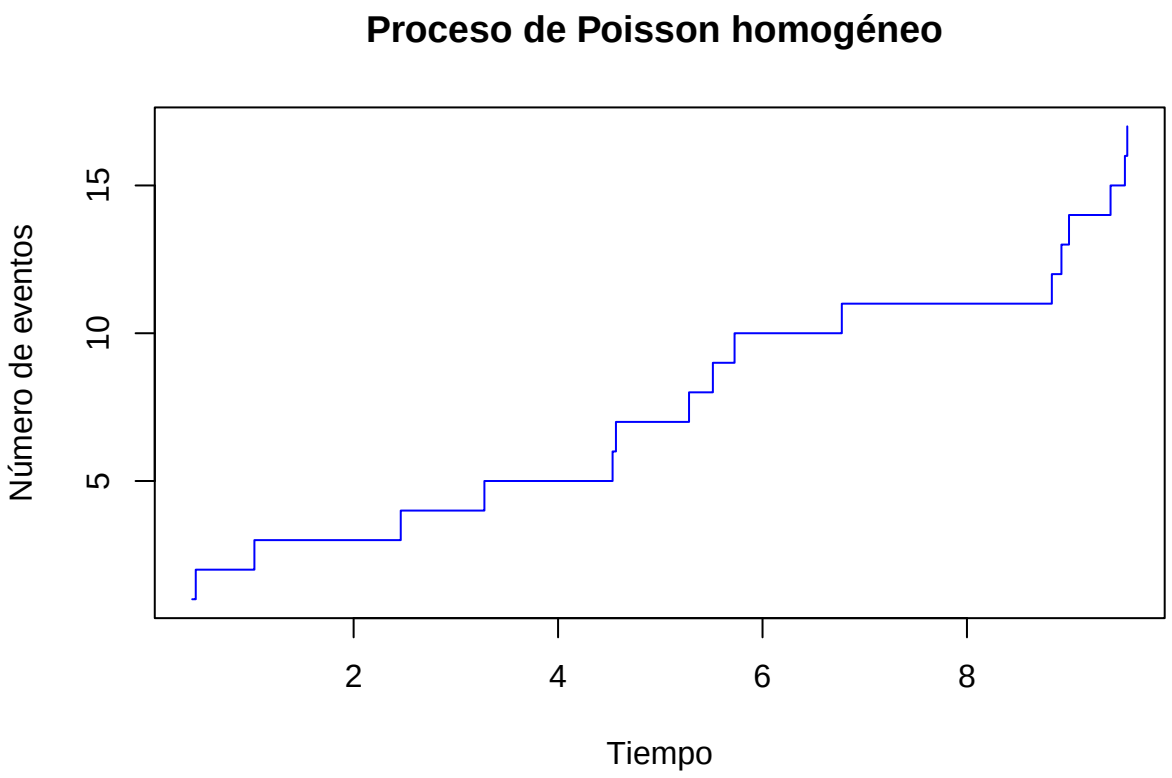
1. $N(0) = 0$.
2. Los incrementos son independientes.
3. El número de eventos en un intervalo de longitud $t$ sigue una Poisson:

$$N(t) \sim \text{Poisson}(\lambda t)$$

**Ejemplo en R: simulación de llegadas**

```r
set.seed(123)
lambda <- 2    # tasa = 2 eventos por unidad de tiempo
Tmax <- 10     # horizonte de tiempo
n <- rpois(1, lambda * Tmax)  # número de eventos
tiempos <- sort(runif(n, 0, Tmax))  # tiempos uniformes

plot(tiempos, 1:n, type="s", col="blue",
     main="Proceso de Poisson homogéneo ",
     xlab="Tiempo", ylab="Número de eventos")
```




2) Proceso de Poisson compuesto

Aquí no solo interesa el número de llegadas, sino también la magnitud de cada evento.

Ejemplo típico: reclamos en seguros

- Llegadas de siniestros $\sim$ Poisson($\lambda t$).
- Montos de siniestros $Y_i$ i.i.d. con cierta distribución.
- El costo total en $[0, t]$ es:

$$S(t) = \sum_{i=1}^{N(t)} Y_i$$

Ejemplo en R: siniestros con montos exponenciales

```r
set.seed(123)
lambda <- 3
Tmax <- 5
n <- rpois(1, lambda * Tmax)   # número de siniestros
tiempos <- sort(runif(n, 0, Tmax))
```

```r
montos <- rexp(n, rate=0.5)    # montos ~ Exp(media=2)

S_total <- sum(montos)
c("Número de siniestros"=n, "Costo total"=S_total)
```

```
## Número de siniestros          Costo total
##             12.00000             21.45492
```

3) Proceso de Poisson no homogéneo

Ahora la tasa $\lambda(t)$ depende del tiempo. La intensidad acumulada es:

$$\Lambda(t) = \int_0^t \lambda(u)\, du$$

y

$$N(t) \sim \text{Poisson}(\Lambda(t))$$

**Ejemplo en R: tasa creciente en el tiempo**

Supongamos $\lambda(t) = 2 + t$ en $[0, 5]$.

$$\Lambda(t) = \int_0^t (2 + u)\, du = 2t + \tfrac{1}{2}t^2$$

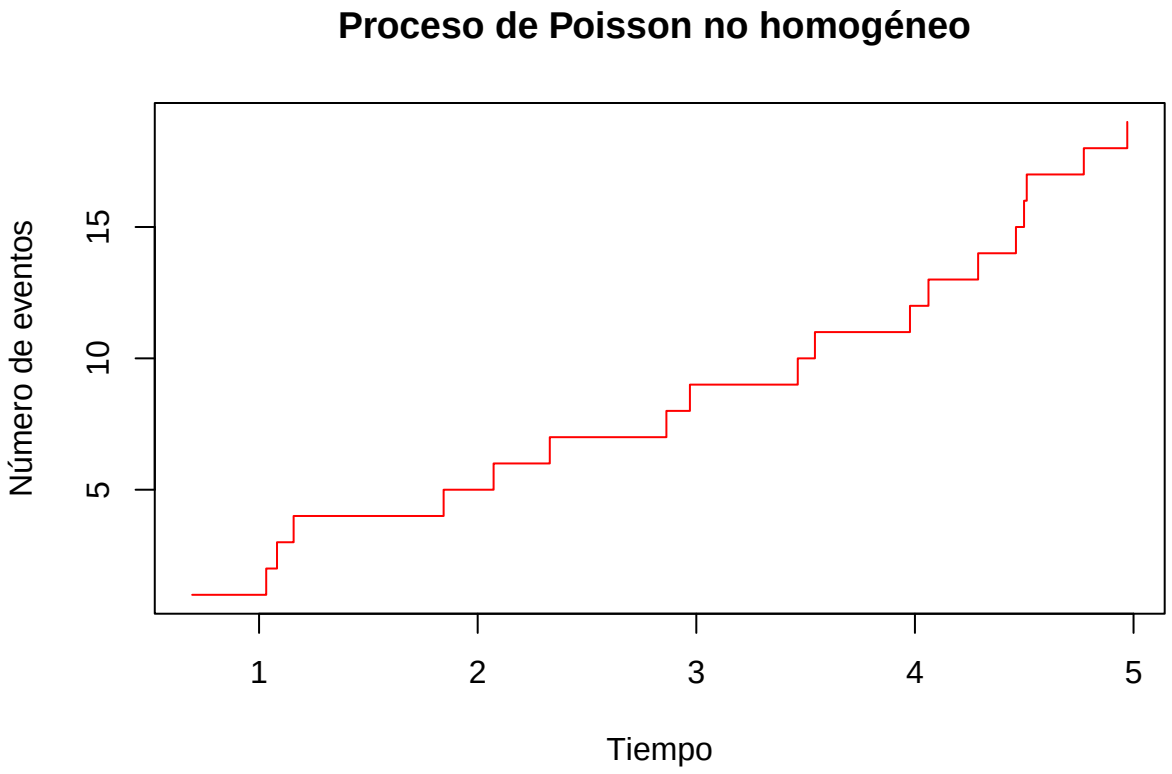
```r
set.seed(123)
Tmax <- 5

# Lambda(Tmax)
Lambda <- function(t) 2*t + 0.5*t^2

# Número de eventos ~ Poisson(Lambda(Tmax))
n <- rpois(1, Lambda(Tmax))

# Método de aceptación-rechazo para generar los tiempos
tiempos <- c()
while(length(tiempos) < n){
  t_cand <- runif(1, 0, Tmax)
  if(runif(1) < (2+t_cand)/max(2+0:Tmax)) {
    tiempos <- c(tiempos, t_cand)
  }
}
tiempos <- sort(tiempos)

plot(tiempos, 1:length(tiempos), type="s", col="red",
     main="Proceso de Poisson no homogéneo ",
     xlab="Tiempo", ylab="Número de eventos")
```


Proceso de Poisson no homogéneo
Número de eventos
5
10
15
1
2
3
4
5
Tiempo

# 3. Aplicaciones y extras

Esta sección tiene como objetivo mostrar algunas herramientas prácticas que complementan lo visto en regresión y procesos estocásticos. Incluye paquetes actuariales muy utilizados en el mundo profesional y una nota final sobre cómo aprovechar ChatGPT como apoyo en el aprendizaje y programación en R.

## 3.1. Paquetes actuariales en R

En ciencias actuariales, existen paquetes especializados que permiten trabajar con seguros de vida, pensiones, tablas de mortalidad y cálculos de riesgo.

**Paquete lifecontingencies**

- Permite trabajar con tablas de mortalidad y seguros de vida.
- Funciones para calcular valores esperados de beneficios, primas netas, valores actuariales.

Ejemplo en R:

```
library(lifecontingencies)
```

```
## Package:  lifecontingencies
## Authors:  Giorgio Alfredo Spedicato [aut, cre]
##     (<https://orcid.org/0000-0002-0315-8888>),
##   Christophe Dutang [ctb] (<https://orcid.org/0000-0001-6732-1501>),
##   Reinhold Kainhofer [ctb] (<https://orcid.org/0000-0002-7895-1311>),
##   Kevin J Owens [ctb],
##   Ernesto Schirmacher [ctb],
##   Gian Paolo Clemente [ctb] (<https://orcid.org/0000-0001-6795-4595>),
##   Ivan Williams [ctb]
## Version:  1.3.12
## Date:     2024-09-29 22:40:06 UTC
## BugReport: https://github.com/spedygiorgio/lifecontingencies/issues
```

```
# Tabla de mortalidad simplificada
data(soa08Act)  # tabla incluida en el paquete

# Seguro de vida temporal: edad 40, n=10 años, i=2%
Axn(soa08Act, x=40, n=10, i=0.02)
```

```
## [1] 0.03453723
```

**Paquete actuar** - Diseñado para modelar pérdidas y riesgos. - Incluye funciones para distribuciones actuariales (Pareto, Burr, etc.). - Herramientas para reaseguro, stop-loss, valores presentes.

Ejemplo en R:

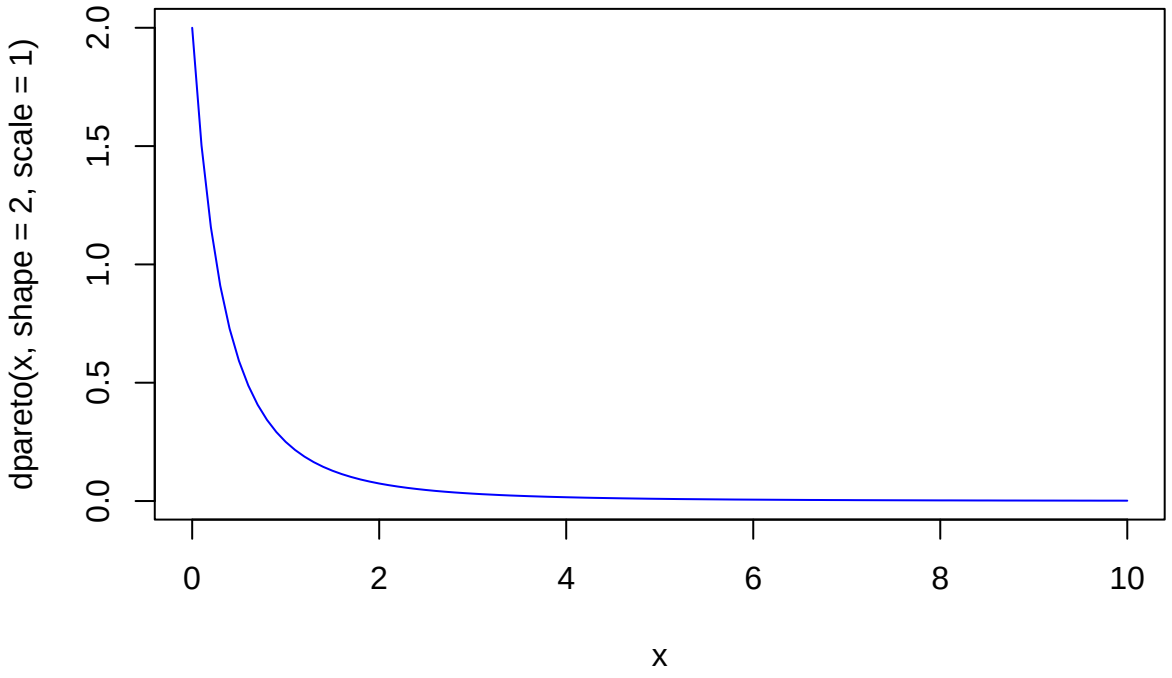
```
library(actuar)
```

```
##
## Attaching package: 'actuar'
```

```
## The following objects are masked from 'package:stats':
##
##     sd, var
```

```
## The following object is masked from 'package:grDevices':
##
##     cm
```

```r
# Distribución Pareto: densidad y simulación
x <- seq(0, 10, by=0.1)
plot(x, dpareto(x, shape=2, scale=1), type="l", col="blue")
```




```r
# Simulación de pérdidas
sim <- rpareto(1000, shape=2, scale=1)
mean(sim); var(sim)
```

```
## [1] 1.016329
```

```
## [1] 6.541536
```

## 3.2. Uso de ChatGPT como apoyo en programación y estadística en R

Hoy en día, herramientas de IA como ChatGPT son un gran aliado para aprender y trabajar en R:

- Apoyo en programación:
  - Depuración de errores de código.
  - Explicación de funciones y paquetes.
  - Generación de ejemplos prácticos.
- Apoyo en estadística y probabilidad:
  - Explicación de conceptos teóricos con ejemplos.
  - Creación de ejercicios personalizados.
  - Interpretación de resultados de salida en R.

Ejemplo práctico de uso en clase:

- Formular en ChatGPT: “Explícame cómo funciona la prueba Breusch–Pagan en R y dame un ejemplo con código reproducible”.
- Recibirás un bloque de código funcional + interpretación.

La IA no reemplaza el criterio profesional. Se debe usar como apoyo, no como sustituto del razonamiento crítico ni de la validación estadística.