Sesión 7 (B) — Verificación geométrica: identidades con datos simulados

Marcos Bujosa

Descripción de la práctica

En las lecciones 4 y 5 demostramos una serie de identidades: el vector en desviaciones es ortogonal a **1**; la varianza satisface una identidad de Pitágoras (fórmula de cálculo); la covarianza es un producto escalar de vectores centrados; la correlación es el coseno del ángulo entre ellos. En la práctica A las verificamos con datos *reales*. Aquí las verificamos con datos *simulados por nosotros mismos*: al controlar por completo cómo se construyen los datos, podemos comprobar que estas identidades no son una peculiaridad de un dataset concreto, sino teoremas que se cumplen siempre, sea cual sea la distribución subyacente o el tamaño de la muestra.

Objetivo

1. Verificar, con datos generados por nosotros, las identidades geométricas de las lecciones 3, 4 y 5.
2. Comprobar que estas identidades no dependen del tamaño muestral ni de la distribución subyacente.
3. Construir explícitamente un caso con una relación lineal conocida entre dos vectores, anticipando el modelo de regresión simple (lección 6).

Actividad 1 - Un vector pequeño, verificación completa

Generamos un vector de solo $n = 6$ observaciones, lo bastante pequeño como para poder mirar los números uno a uno.

en línea de comandos:

```
nulldata 6 --preserve
set seed 12345

series y = normal(10, 3)
print y
```

Ortogonalidad del vector en desviaciones

en línea de comandos:

```
printf "Suma de (y - mu_y) = %.10f   (debe ser 0)\n", sum(y - mean(y))
```

```
Suma de (y - mu_y) = -0,0000000000   (debe ser 0)
```

Pitágoras (fórmula de cálculo de la varianza)

Recuerde la identidad de la lección 5: $\mu_{\boldsymbol{y}^2} = \mu_{\boldsymbol{y}}^2 + \sigma_{\boldsymbol{y}}^2$, donde $\sigma_{\boldsymbol{y}}^2$ es la varianza tal como la define la lección 5: con divisor $1/n$, el del producto escalar estadístico.

en línea de comandos:

```
scalar mu_y      = mean(y)
scalar var_n     = sum((y - mu_y)^2) / $nobs        # divisor 1/n (convención del curso)
scalar mu_y2     = mean(y^2)

printf "mu_y2                = %.6f\n", mu_y2
printf "mu_y^2 + sigma_y^2   = %.6f   (divisor n)\n", mu_y^2 + var_n
```

```
mu_y2                = 149,479709
mu_y^2 + sigma_y^2   = 149,479709   (divisor n)
```

IMPORTANTE — a diferencia de la correlación (Apéndice de la práctica A), aquí el divisor SÍ importa: la identidad de Pitágoras usa el producto escalar estadístico con divisor $1/n$ (lección 5). Compruebe qué ocurre si, en su lugar, divide por $(n-1)$ (el convenio muestral habitual en estadística inferencial, y el que usa por defecto la función/=var= de Gretl):/

```
scalar var_n1 = sum((y - mu_y)^2) / ($nobs - 1)     # divisor 1/(n-1)
printf "mu_y^2 + var(n-1)    = %.6f   (divisor n-1, ya NO coincide con mu_y2)\n", mu_y^2 + var_n1
```

```
mu_y^2 + var(n-1)    = 153,621305   (divisor n-1, ya NO coincide con mu_y2)
```

A diferencia del coseno (un cociente, invariante al divisor), la varianza es una magnitud absoluta: cambiar el divisor SÍ cambia su valor numérico. La identidad de Pitágoras de la lección 5 se demostró con el divisor $1/n$; es EXACTA con ese convenio, y solo aproximada (para n grande) con $(n-1)$.

Actividad 2 - ¿Depende del tamaño o de la distribución?

Repetimos las mismas comprobaciones (ortogonalidad y Pitágoras) con dos series de $n = 50$: una con forma simétrica (normal) y otra con forma marcadamente asimétrica (χ^2 con 2 grados de libertad, igual que la Serie B de la práctica C de esta misma sesión). Obsérvese que con esto cambian *las dos cosas* respecto a la Actividad 1: allí el tamaño era $n = 6$, aquí es $n = 50$ y en la Actividad 3 será $n = 100$; entre las tres actividades se recorren, pues, tres tamaños muestrales muy distintos además de tres formas distintas. Lo hacemos primero "a mano", serie a serie, para que quede completamente explícito qué se calcula en cada caso.

en línea de comandos:

```
nulldata 50 --preserve
set seed 54321

series y_normal = normal(0, 1)
series u1 = normal(0,1)
series u2 = normal(0,1)
series y_chi2 = u1^2 + u2^2      # marcadamente asimétrica

# --- y_normal ---
scalar mu_normal    = mean(y_normal)
scalar var_normal   = sum((y_normal - mu_normal)^2) / $nobs
scalar ortog_normal = sum(y_normal - mu_normal)
scalar pitag_normal = mean(y_normal^2) - (mu_normal^2 + var_normal)
printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "y_normal", ortog_normal, pitag_normal

# --- y_chi2 ---
scalar mu_chi2    = mean(y_chi2)
scalar var_chi2   = sum((y_chi2 - mu_chi2)^2) / $nobs
scalar ortog_chi2 = sum(y_chi2 - mu_chi2)
scalar pitag_chi2 = mean(y_chi2^2) - (mu_chi2^2 + var_chi2)
printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "y_chi2", ortog_chi2, pitag_chi2
```

¿Cambia algo el resultado por tratarse de una distribución simétrica o muy asimétrica? ¿Por qué no?

```
y_normal   -> ortogonalidad: 0,0000000000    Pitágoras (desajuste): 0,0000000000
y_chi2     -> ortogonalidad: 0,0000000000    Pitágoras (desajuste): 0,0000000000
```

Una variante más compacta (opcional): el mismo cálculo con `loop`

Repetir a mano el mismo bloque de código para cada serie es razonable con dos series, pero se volvería tedioso con veinte. Gretl ofrece el comando `loop` precisamente para automatizar este tipo de repeticiones. Al igual que ocurrirá en la Actividad 5, no pretendemos aquí explicar la mecánica de los bucles (`loop`) de Gretl —eso se hará con detalle en el laboratorio que sigue a la lección 11 (propiedades de MCO) y en futuras prácticas sobre inferencia (contrastes t y F)—; aquí lo presentamos simplemente como una "caja negra" que aplica, serie a serie, el mismo cálculo que acabamos de hacer a mano. La ventaja se aprecia enseguida: con muy poco código adicional podríamos repetir esta misma comprobación sobre diez, veinte o cien series distintas, sin copiar el bloque una y otra vez.

en línea de comandos (variante con `loop`, mismo resultado que arriba):

```
loop foreach i y_normal y_chi2 --quiet
    scalar mu    = mean($i)
    scalar var_n = sum(($i - mu)^2) / $nobs
    scalar ortog = sum($i - mu)
    scalar pitag = mean($i^2) - (mu^2 + var_n)
    printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "$i", ortog, pitag
endloop
```

```
y_normal   -> ortogonalidad: 0,0000000000    Pitágoras (desajuste): 0,0000000000
y_chi2     -> ortogonalidad: 0,0000000000    Pitágoras (desajuste): 0,0000000000
```

Un caso extremo: con $n = 2$ la correlación es siempre ± 1

La lección 5 demostró un hecho que conviene ver con los propios ojos. En $\mathbb{R}^2$, el complemento ortogonal $\mathcal{L}(\mathbf{1})^{\perp}$ es una simple *recta*; y dos vectores cualesquiera situados sobre una recta están forzosamente alineados. Por tanto, con solo dos observaciones los vectores en desviaciones siempre están alineados y la correlación vale ± 1, *sean cuales sean los datos.*

en línea de comandos:

```
nulldata 2 --preserve
set seed 777
series a = normal()
series b = normal()
printf "Primer sorteo  (n=2): corr(a,b) = %+.10f\n", corr(a,b)

set seed 778
series a = normal()
series b = normal()
printf "Segundo sorteo (n=2): corr(a,b) = %+.10f\n", corr(a,b)
```

```
Primer sorteo  (n=2): corr(a,b) = +1,0000000000
Segundo sorteo (n=2): corr(a,b) = -1,0000000000
```

Los dos sorteos son independientes y dan datos completamente distintos, pero la correlación sale ± 1 en ambos. No es casualidad ni redondeo: con $n = 2$ no hay ninguna otra posibilidad. La consecuencia práctica es importante: una correlación de 1 calculada con dos observaciones no informa de nada. Para observar correlaciones intermedias hacen falta al menos tres datos, que es la razón por la que las figuras de la lección 5 están dibujadas en $\mathbb{R}^3$.

Actividad 3 - Dos vectores con relación conocida (puente a la regresión)

Construimos ahora $\boldsymbol{x}$ y $\boldsymbol{y}$ *a propósito*, de forma que conocemos exactamente cómo se relacionan: $\boldsymbol{y} = 2+3\boldsymbol{x}+\boldsymbol{u}$, con $\boldsymbol{u}$ un término de ruido generado con independencia de $\boldsymbol{x}$ y, por tanto, prácticamente incorrelado con él.

en línea de comandos:

```
nulldata 100 --preserve
set seed 11223

series x = uniform(0, 10)
series u = normal(0, 2)
series y = 2 + 3*x + u
```

Atención: $\boldsymbol{u}$ es el vector con el que *hemos construido* $\boldsymbol{y}$ —lo conocemos porque lo hemos fabricado nosotros—, no algo que pueda leerse en los datos. Cuando en la lección 6 aparezca el vector de error de un ajuste, será un objeto distinto: lo que queda al restar a $\boldsymbol{y}$ su ajuste, calculado a partir de $\boldsymbol{x}$ e $\boldsymbol{y}$ y nada más. Conviene no confundirlos; volveremos sobre ello cuando tengamos el segundo.

Covarianza como producto escalar de vectores centrados

en línea de comandos:

```
scalar x_c_dot_y_c = sum((x - mean(x)) * (y - mean(y)))
scalar cov_manual  = x_c_dot_y_c / $nobs

printf "Covarianza \"a mano\" (divisor n) = %.6f\n", cov_manual
printf "cov(x,y) de Gretl (divisor n-1)  = %.6f\n", cov(x,y)
```

```
Covarianza "a mano" (divisor n) = 26,372167
cov(x,y) de Gretl (divisor n-1)  = 26,638553
```

Correlación como coseno

en línea de comandos:

```
scalar norma_x = sqrt(sum((x - mean(x))^2))
scalar norma_y = sqrt(sum((y - mean(y))^2))
scalar coseno  = x_c_dot_y_c / (norma_x * norma_y)

printf "cos(theta) \"a mano\" = %.6f\n", coseno
printf "corr(x,y) de Gretl  = %.6f\n", corr(x,y)
```

```
cos(theta) "a mano" = 0,972574
corr(x,y) de Gretl  = 0,972574
```

¿Qué valor obtendremos para la correlación si construimos $\boldsymbol{y}$ *con la fórmula* $\boldsymbol{y} = 2 + 3\boldsymbol{x}$*? (es decir, sin* $\boldsymbol{u}$*).*

Aquí conocemos la "verdad": $\boldsymbol{y}$ *se construyó con pendiente 3 respecto a* $\boldsymbol{x}$*. En las dos próximas lecciones veremos cómo* recuperar *esa pendiente a partir solo de los datos observados, sin conocerla de antemano: la lección 6 plantea el problema y la lección 7 lo resuelve (estimación MCO).*

Cauchy–Schwarz y la desigualdad triangular

La lección 3 demostró dos desigualdades que hasta ahora no habíamos comprobado con datos. Las dos se leen directamente sobre los vectores en desviaciones que acabamos de construir.

en línea de comandos:

```
scalar norma_suma = sqrt(sum(((x - mean(x)) + (y - mean(y)))^2))

printf "Cauchy-Schwarz:  |<x_c,y_c>| = %.4f  <=  ||x_c||*||y_c|| = %.4f\n", \
       abs(x_c_dot_y_c), norma_x * norma_y
printf "Triangular:      ||x_c+y_c|| = %.4f  <=  ||x_c||+||y_c|| = %.4f\n", \
       norma_suma, norma_x + norma_y
```

```
Cauchy-Schwarz:  |<x_c,y_c>| = 2637,2167  <=  ||x_c||*||y_c|| = 2711,5839
Triangular:      ||x_c+y_c|| = 119,1665  <=  ||x_c||+||y_c|| = 119,7889
```

La primera desigualdad es la que sostiene todo lo demás: dividiendo ambos lados por $\|x_c\|\|y_c\|$ se obtiene $|\rho| \leq 1$, es decir, que el coseno que hemos calculado nunca puede salirse del intervalo $[-1, 1]$. Sin Cauchy–Schwarz, llamar "coseno" a ese cociente en $\mathbb{R}^n$ no estaría justificado. Nótese además que ambas desigualdades son estrictas aquí, porque $\boldsymbol{x}$ e $\boldsymbol{y}$ no están alineados; la igualdad solo se alcanza cuando uno es múltiplo del otro —justo el caso que acaba de ver con $n = 2$—.

Actividad 4 - Síntesis

Estas identidades y desigualdades —ortogonalidad, Pitágoras, covarianza como producto escalar, correlación como coseno, Cauchy–Schwarz y la desigualdad triangular— son *teoremas* de la geometría de $\mathbb{R}^n$ (lecciones 3, 4 y 5), no propiedades empíricas de un dataset concreto. Por eso se cumplen igual con datos reales (práctica A) que con datos simulados de cualquier distribución y cualquier tamaño (aquí). La próxima sesión da un paso más: en lugar de *verificar* identidades que ya conocemos, empezaremos a *usarlas* para construir un estimador (MCO), camino que se completa en la lección 7.

Actividad 5 (OPCIONAL, según el tiempo disponible y las preguntas) - ¿Y si repetimos esto cientos de veces?

Esta actividad es opcional: es probable que no dé tiempo a cubrirla en clase. Está reservada para si surge la pregunta natural: "¿esto se cumple siempre, o ha sido casualidad de esta muestra concreta?" No se pretende aquí explicar la mecánica de los bucles (`loop`) de Gretl –eso se hará con detalle en el laboratorio que sigue a la lección 11 (propiedades de MCO) y en las prácticas sobre inferencia (contrastes t y F)–; aquí se presenta como una "caja negra" que repite la verificación cientos de veces con una muestra nueva en cada repetición.

en línea de comandos:

```
nulldata 30 --preserve
set seed 987654

scalar R = 500
scalar max_dev_ortog = 0
scalar max_dev_pitag = 0
scalar max_dev_rho   = 0

loop R --quiet
    series y_loop = normal(0,1) + (uniform(0,1) < 0.1) * normal(5,1)
    series x_loop = uniform(0,10)

    scalar mu_loop   = mean(y_loop)
    scalar var_loop  = sum((y_loop - mu_loop)^2) / $nobs

    scalar dev_ortog = abs(sum(y_loop - mu_loop))
    scalar dev_pitag = abs(mean(y_loop^2) - (mu_loop^2 + var_loop))

    scalar dp_loop    = sum((x_loop-mean(x_loop))*(y_loop-mu_loop))
    scalar norma_xl   = sqrt(sum((x_loop-mean(x_loop))^2))
```

```
    scalar norma_yl   = sqrt(sum((y_loop-mu_loop)^2))
    scalar coseno_l   = dp_loop / (norma_xl*norma_yl)
    scalar dev_rho    = abs(coseno_l - corr(x_loop,y_loop))

    if dev_ortog > max_dev_ortog
        max_dev_ortog = dev_ortog
    endif
    if dev_pitag > max_dev_pitag
        max_dev_pitag = dev_pitag
    endif
    if dev_rho > max_dev_rho
        max_dev_rho = dev_rho
    endif
endloop

printf "\nTras %d repeticiones, cada una con una muestra nueva:\n", R
printf "  Desviación MÁXIMA ortogonalidad    : %.10f\n", max_dev_ortog
printf "  Desviación MÁXIMA Pitágoras        : %.10f\n", max_dev_pitag
printf "  Desviación MÁXIMA rho = cos(theta) : %.10f\n", max_dev_rho
```

```
Tras 500 repeticiones, cada una con una muestra nueva:
  Desviación MÁXIMA ortogonalidad    : 0,0000000000
  Desviación MÁXIMA Pitágoras        : 0,0000000000
  Desviación MÁXIMA rho = cos(theta) : 0,0000000000
```

Las tres desviaciones máximas deberían ser minúsculas (por errores de redondeo de Gretl) o directamente cero. Esto ilustra que no importa cuántos *ejemplos distintos se generen: las identidades geométricas se cumplen siempre.*

Preguntas de interpretación para la clase

1. ¿Por qué el resultado de la Actividad 2 no depende de si la distribución es simétrica o muy asimétrica?
2. En la Actividad 1, ¿por qué la identidad de Pitágoras deja de cumplirse exactamente al dividir por $(n-1)$ en vez de por n? ¿Es esto una contradicción con la invarianza de la correlación vista en la práctica A?
3. En la Actividad 3, ¿qué relación hay entre el signo de la pendiente conocida (3, positiva) y el signo de la correlación obtenida?
4. ¿Qué diferencia hay entre "verificar" una identidad (lo que hacemos en esta práctica) y "demostrarla" (lo que se hizo en las lecciones 3, 4 y 5)?
5. (Si se trabajó la Actividad 5) Tras 500 repeticiones con muestras distintas cada vez, ¿por qué la desviación máxima observada sigue siendo del mismo orden de magnitud que en una sola muestra, en vez de ir creciendo con el número de repeticiones?

Para profundizar:

- Véanse las lecciones 3, 4 y 5 para las demostraciones completas de lo que aquí se verifica numéricamente.
- Cottrell, A. y Lucchetti, R. (2023). *Gretl User's Guide*. Funciones de generación aleatoria (`normal`, `uniform`), `set seed`, y estructura básica del comando `loop` (consulta puntual; su uso sistemático se explica en el laboratorio que sigue a la lección 11).

Código completo de la práctica

```
# ----------------------------------------------------------
# Copyright (C) 2026 Marcos Bujosa
# Licencia: GNU General Public License v3.0 o posterior
# Este código es software libre y puede ser redistribuido y/o modificado bajo los términos de la GPL.
# Ver el archivo LICENSE del repositorio para más detalles.
# ----------------------------------------------------------

# Los dos primeros comandos son necesarios para que Gretl guarde los resultados de la práctica en el
↪  directorio de trabajo
# al ejecutar lo siguiente desde un terminal (use los nombres y ruta que correspondan)
#
#    DIRECTORIO="Nombre_Directorio_trabajo" gretlcli -b ruta/nombre_fichero_de_la_practica.inp
#
# Si esto no le funciona en su sistema, comente las siguientes dos líneas y sitúese en el directorio de
↪  trabajo de gretl
# que corresponda (configure dicho directorio de trabajo desde la ventana principal de Gretl).

string directory = getenv("DIRECTORIO")
set workdir "@directory"

nulldata 6 --preserve
set seed 12345

series y = normal(10, 3)
print y

outfile --quiet ortogonalidad.txt
    printf "Suma de (y - mu_y) = %.10f   (debe ser 0)\n", sum(y - mean(y))
end outfile

outfile --quiet ortogonalidad-n.txt
    scalar mu_y      = mean(y)
    scalar var_n     = sum((y - mu_y)^2) / $nobs        # divisor 1/n (convención del curso)
    scalar mu_y2     = mean(y^2)

    printf "mu_y2                = %.6f\n", mu_y2
    printf "mu_y^2 + sigma_y^2   = %.6f   (divisor n)\n", mu_y^2 + var_n
end outfile

outfile --quiet ortogonalidad-n-1.txt
    scalar var_n1 = sum((y - mu_y)^2) / ($nobs - 1)     # divisor 1/(n-1)
    printf "mu_y^2 + var(n-1)    = %.6f   (divisor n-1, ya NO coincide con mu_y2)\n", mu_y^2 + var_n1
end outfile

outfile --quiet dosSeriesn50amano.txt
    nulldata 50 --preserve
    set seed 54321

    series y_normal = normal(0, 1)
    series u1 = normal(0,1)
    series u2 = normal(0,1)
    series y_chi2 = u1^2 + u2^2      # marcadamente asimétrica

    # --- y_normal ---
    scalar mu_normal    = mean(y_normal)
    scalar var_normal   = sum((y_normal - mu_normal)^2) / $nobs
    scalar ortog_normal = sum(y_normal - mu_normal)
    scalar pitag_normal = mean(y_normal^2) - (mu_normal^2 + var_normal)
    printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "y_normal", ortog_normal,
    ↪  pitag_normal

    # --- y_chi2 ---
    scalar mu_chi2    = mean(y_chi2)
    scalar var_chi2   = sum((y_chi2 - mu_chi2)^2) / $nobs
```

```
    scalar ortog_chi2 = sum(y_chi2 - mu_chi2)
    scalar pitag_chi2 = mean(y_chi2^2) - (mu_chi2^2 + var_chi2)
    printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "y_chi2", ortog_chi2,
    ↪  pitag_chi2
end outfile

outfile --quiet dosSeriesn50conloop.txt
    loop foreach i y_normal y_chi2 --quiet
        scalar mu    = mean($i)
        scalar var_n = sum(($i - mu)^2) / $nobs
        scalar ortog = sum($i - mu)
        scalar pitag = mean($i^2) - (mu^2 + var_n)
        printf "%-10s -> ortogonalidad: %.10f    Pitágoras (desajuste): %.10f\n", "$i", ortog, pitag
    endloop
end outfile

outfile --quiet correlacionN2.txt
    nulldata 2 --preserve
    set seed 777
    series a = normal()
    series b = normal()
    printf "Primer sorteo  (n=2): corr(a,b) = %+.10f\n", corr(a,b)

    set seed 778
    series a = normal()
    series b = normal()
    printf "Segundo sorteo (n=2): corr(a,b) = %+.10f\n", corr(a,b)
end outfile

nulldata 100 --preserve
set seed 11223

series x = uniform(0, 10)
series u = normal(0, 2)
series y = 2 + 3*x + u

outfile --quiet covarianzamanual.txt
    scalar x_c_dot_y_c = sum((x - mean(x)) * (y - mean(y)))
    scalar cov_manual  = x_c_dot_y_c / $nobs

    printf "Covarianza \"a mano\" (divisor n) = %.6f\n", cov_manual
    printf "cov(x,y) de Gretl (divisor n-1)  = %.6f\n", cov(x,y)
end outfile

scalar norma_x = sqrt(sum((x - mean(x))^2))
scalar norma_y = sqrt(sum((y - mean(y))^2))
scalar coseno  = x_c_dot_y_c / (norma_x * norma_y)

printf "cos(theta) \"a mano\" = %.6f\n", coseno
printf "corr(x,y) de Gretl  = %.6f\n", corr(x,y)

outfile --quiet correlacioncomocosenoxy.txt
    scalar norma_x = sqrt(sum((x - mean(x))^2))
    scalar norma_y = sqrt(sum((y - mean(y))^2))
    scalar coseno  = x_c_dot_y_c / (norma_x * norma_y)

    printf "cos(theta) \"a mano\" = %.6f\n", coseno
    printf "corr(x,y) de Gretl  = %.6f\n", corr(x,y)
end outfile

outfile --quiet desigualdades.txt
    scalar norma_suma = sqrt(sum(((x - mean(x)) + (y - mean(y)))^2))

    printf "Cauchy-Schwarz:  |<x_c,y_c>| = %.4f  <=  ||x_c||*||y_c|| = %.4f\n", \
           abs(x_c_dot_y_c), norma_x * norma_y
    printf "Triangular:      ||x_c+y_c|| = %.4f  <=  ||x_c||+||y_c|| = %.4f\n", \
           norma_suma, norma_x + norma_y
```

```
end outfile

outfile --quiet loopopcionalderepeticionmasiva.txt
    nulldata 30 --preserve
    set seed 987654

    scalar R = 500
    scalar max_dev_ortog = 0
    scalar max_dev_pitag = 0
    scalar max_dev_rho   = 0

    loop R --quiet
        series y_loop = normal(0,1) + (uniform(0,1) < 0.1) * normal(5,1)
        series x_loop = uniform(0,10)

        scalar mu_loop   = mean(y_loop)
        scalar var_loop  = sum((y_loop - mu_loop)^2) / $nobs

        scalar dev_ortog = abs(sum(y_loop - mu_loop))
        scalar dev_pitag = abs(mean(y_loop^2) - (mu_loop^2 + var_loop))

        scalar dp_loop    = sum((x_loop-mean(x_loop))*(y_loop-mu_loop))
        scalar norma_xl   = sqrt(sum((x_loop-mean(x_loop))^2))
        scalar norma_yl   = sqrt(sum((y_loop-mu_loop)^2))
        scalar coseno_l   = dp_loop / (norma_xl*norma_yl)
        scalar dev_rho    = abs(coseno_l - corr(x_loop,y_loop))

        if dev_ortog > max_dev_ortog
            max_dev_ortog = dev_ortog
        endif
        if dev_pitag > max_dev_pitag
            max_dev_pitag = dev_pitag
        endif
        if dev_rho > max_dev_rho
            max_dev_rho = dev_rho
        endif
    endloop

    printf "\nTras %d repeticiones, cada una con una muestra nueva:\n", R
    printf "  Desviación MÁXIMA ortogonalidad    : %.10f\n", max_dev_ortog
    printf "  Desviación MÁXIMA Pitágoras        : %.10f\n", max_dev_pitag
    printf "  Desviación MÁXIMA rho = cos(theta) : %.10f\n", max_dev_rho
end outfile
```

Enlace al guión: S07-Prct-B-verificacion-geometrica.inp

Respuestas

1. ¿Por qué el resultado de la Actividad 2 no depende de si la distribución es simétrica o muy asimétrica? Porque la ortogonalidad del vector en desviaciones ($\sum(y_i - \mu_{\boldsymbol{y}}) = 0$) y la identidad de Pitágoras ($\mu_{\boldsymbol{y}^2} = \mu_{\boldsymbol{y}}^2 + \sigma_{\boldsymbol{y}}^2$) son consecuencias *algebraicas* de cómo se define la media (como proyección) y la varianza (como norma al cuadrado del vector en desviaciones): se demuestran manipulando sumas y productos escalares, sin usar en ningún momento supuestos sobre la forma de la distribución (normalidad, simetría, etc.). Son propiedades de *cualquier* vector de $\mathbb{R}^n$, no de un tipo particular de dato. Por eso da igual que $\boldsymbol{y}$ sea normal o χ^2: la demostración de la lección 5 no distingue casos.

2. ¿Por qué Pitágoras deja de cumplirse exactamente al dividir por $(n-1)$? ¿Contradice la invarianza de ρ? No hay contradicción: son cosas distintas. La correlación es un *cociente* (covarianza entre producto de dos "normas"), y en un cociente cualquier factor común al numerador y al denominador se cancela —por eso da igual n, $n-1$, o ningún divisor—. La identidad de Pitágoras, en cambio, es una *igualdad entre magnitudes absolutas* ($\mu_{\boldsymbol{y}^2}$ por un lado, $\mu_{\boldsymbol{y}}^2 + \sigma_{\boldsymbol{y}}^2$ por otro): aquí el divisor no se cancela, porque solo aparece en un término ($\sigma_{\boldsymbol{y}}^2$) y no en el otro ($\mu_{\boldsymbol{y}^2}$, que no lleva divisor de varianza). La demostración de la lección 5 reutiliza el hecho $\|\mathbf{1}\|_s = 1$ —establecido en la lección 4—, que solo es cierto con el producto escalar estadístico (divisor $1/n$); con $(n-1)$ esa pieza deja de encajar exactamente, y la identidad solo se cumple de forma aproximada (cada vez mejor cuanto mayor sea n, porque $n/(n-1) \to 1$).

3. Relación entre el signo de la pendiente conocida (3, positiva) y el signo de la correlación obtenida. Deben coincidir: si $\boldsymbol{y} = 2 + 3\boldsymbol{x} + \boldsymbol{u}$ con $\boldsymbol{u}$ independiente de $\boldsymbol{x}$, entonces $\sigma_{\boldsymbol{xy}} = 3\,\sigma_{\boldsymbol{x}}^2 + \sigma_{\boldsymbol{xu}}$, y como $\boldsymbol{u}$ se generó con independencia de $\boldsymbol{x}$, su covarianza con $\boldsymbol{x}$ es (aproximadamente) cero; queda $\sigma_{\boldsymbol{xy}} \approx 3\,\sigma_{\boldsymbol{x}}^2 > 0$. Como la varianza es siempre positiva, el signo de la covarianza —y por tanto de ρ, que es esa covarianza reescalada por normas positivas— es el signo de la pendiente, aquí positivo.

4. Diferencia entre "verificar" y "demostrar" una identidad. Demostrar es un argumento *general*, válido para cualquier vector de $\mathbb{R}^n$, basado únicamente en las propiedades del producto escalar y la norma (lecciones 3, 4 y 5): una vez hecho, sabemos que la identidad se cumple *siempre*, sin excepción. Verificar es comprobar numéricamente que la identidad se cumple en un *caso concreto* (un vector particular, real o simulado): por sí sola, una verificación aislada no prueba nada en general —podría ser casualidad de esos datos concretos—, pero sirve para (a) ganar confianza intuitiva, (b) detectar errores de cálculo o de comprensión, y (c) mostrar que la demostración abstracta efectivamente "funciona" en la práctica.

5. (Actividad 5, opcional) Tras 500 repeticiones, ¿por qué la desviación máxima no crece con el número de repeticiones? Porque cada repetición es una comprobación *independiente* de la misma identidad algebraica exacta, no una acumulación de error: en cada una de las 500 muestras nuevas, la ortogonalidad y Pitágoras se cumplen (salvo error de redondeo de la máquina, siempre del mismo orden diminuto, 10^{-10} o menor). No hay ningún mecanismo por el que los errores de una repetición se "sumen" a los de la siguiente —son 500 verificaciones *numéricas* independientes de un teorema, no un proceso acumulativo—. Esto contrasta con lo que veremos en el laboratorio que sigue a la lección 11: allí el `loop` no verificará una identidad exacta, sino que estimará una *distribución* (la de $\hat{\beta}_2$ entre todas las muestras), y ahí sí el número de repeticiones importa, porque cuantas más repeticiones, mejor se aproxima esa distribución empírica a la teórica.